www.fre2014.uni-hohenheim.de

Field Robot Event

12th edition

Proceedings

Proceedings of the 12th Field Robot Event 2014

Bernburg-Strenzfeld/Germany
June 17th – 19th, 2014

Conducted in conjunction with the DLG-Feldtage / DLG Field Days

Editors:
M.Sc. Manuel Vázquez Arellano
Prof. Dr. Hans W. Griepentrog

Date: April 2015
Publisher: University of Hohenheim, Instrumentation & Test Engineering (440c)
Stuttgart, Germany
Contact: Prof. Dr. Hans W. Griepentrog
Phone: +49-(0)711-459-24550

File available on this webpage: www.fre2014.uni-hohenheim.de/

Index of Robots

1. Sponsors


DLG


CLAAS
Stiftung


EurAgEng


Hochschule Anhalt
Anhalt University of Applied Sciences


BOSCH


JOHN DEERE


HORSCH


KRONE


BAYER
Bayer


geo-konzept
inventarisieren · kartieren · optimieren


LINAK®
WE IMPROVE YOUR LIFE

2. Task Description

Together with the DLG-Feldtage, 17th – 19th June 2014 at the
International DLG Crop Production Center (IPZ), Bernburg-Strenzfeld, Germany

Remark: We tried to describe the tasks and assessments as good and fair as possible, but all teams should be aware of that we might need to modify the rules before or even during the contest! These ad hoc changes will always be discussed and decided by the jury.

0. Introduction

During the last years the conducted tasks were always related to crop row structures. Again in 2014 we will focus on crop rows. After having sunflowers and rose pots during last two years we will return to maize[1] in 2014 as used already during several previous FRE contests.

Five tasks will be prepared to challenge different abilities of the robots in terms of sensing, navigation and actuation. Traditionally as well as this year task 1 (basic navigation) and task 2 (advanced navigation) require proper sensing in order to achieve accurate and fast navigation between crop rows. In task 3 the agricultural application will be visible by letting the robots detect weed plants and create weed maps using positional information from a GNSS.

For task 4 always two teams will be asked to let their robots work together to show cooperative abilities. With regards to last contests the team or robot cooperation was highly appreciated. In 2014 we will conduct the popular discipline Freestyle as task 5.

In task 4 and 5 the teams are totally free in to present a robot performance based on their own innovative ideas. As during previous contests the general use of a GNSS system is NOT allowed, because the focus shall be on relative positioning and sensor based behaviours. However, in 2014 we will use them in task 3 for weed mapping (absolute positioning) and on team request in task 4 (Collaboration) and 5 (Freestyle).

All participating teams must contribute to the contest proceedings with an article describing the machine (mechanics and hard- & software) and perhaps their ideas behind or development strategies.

0.1. General rules

All machines shall operate in autonomous mode. Therefore, to control or guide them with laptops, specific controllers or other devices is not allowed. Furthermore, no remote hardware or data sources are allowed, only machine on-board systems shall be used. However, one person is allowed to follow the

[1] Plant density 10 m^{-2}, row width of 0.75 m, plant spacing 0.133 m

machine to correct it in case undesired performance or when an emergency stop is needed.

During the contests all robots have to wait in the parc fermé and no more machine modification to change the machine performance is - with regard to fairness - allowed. All PC connections (wired and wireless) have to be removed or switched off and an activation of a battery saving mode is recommended. This shall avoid having an advantage not being the first robot to conduct the task. The starting order will be random. When the 1st robot will move to the starting point the next robot will already be asked by the parc fermé officer to prepare for starting.

At the starting point the robot must start within one minute. If the robot doesn't start within this time, it will get a second chance after all other teams finished their runs, but it must - after a basic repair - as soon as possible brought back into the parc ferme. If the robot fails twice, the robot will be excluded for that task.

For task 3 and on request for task 4 and task 5 two battery powered GNSS boxes including antennas will be provided by the organiser. The specifications will be published on the web pages in advance.

The drive paths of the robots shall be between the crop rows and not above rows. Large robots or robots which probably partly damage the field or plants will always start after the other robots, including the second chance starting robots. However, damaged plants will be replaced by spare ones to ensure always the same operation conditions for each run.

0.2. Awards

The performance of the competing robots will be assessed by an independent expert jury. Beside measured or counted performance parameters also creativity and originality especially in tasks 4 (Collaboration) and task 5 (Freestyle) will be evaluated. There will be an award for the first three ranks of each task. The basic (1), advanced (2) and professional task (3) together will yield the overall competition winner. Points will be given as follows:

Rank	1	2	3	4	5	6	7	8	9	etc.
Points	10	8	6	5	4	3	2	1	1	etc.

Participating teams result in at least 1 point, not participating teams result in 0 points. If two or more teams have the same number of points for the overall ranking, the team with the better placements during all three tasks (1, 2 and 3) will be ranked higher.

1. Task “Basic navigation” (1)

1.1. General description

Within three minutes the robot has to navigate through long curved rows of maize plants (*picture 1* at the bottom of this text). The aim is to cover as much distance as possible. On the headland, the robot has to turn and return in the adjacent row. There will be no plants missing in the rows. This task is all about accuracy, smoothness and speed of the navigation operation between the rows.

At the beginning of the match it will be told whether starting is on the left side of the field (first turn is right) or on the right side (first turn is left). This is not a choice of the team but of the officials. Therefore, the robots should be able to perform for both options. A headland width of 1.5 meters free of obstacles (bare soil) will be available for turning.

1.2. Assessment

The distance travelled in 3 minutes is measured. If the end of the field is reached in less time, this actually used time will be used to calculate a

bonus factor = total distance * 3 minutes / measured time.

The total distance includes travelled distance and the penalty values. Distance and time are measured by the jury officials.

Manual intervention during the within field operation will result in a distance penalty of 3 meter per touch. During headland turning – after exiting the rows - to help the robot finding the right row will be punished with a penalty of 5 meters. The number of interventions (touches) will be counted by the officials.

Crop plant damage by the robot (e.g. bended, broken or uprooted plants) will result in a penalty of 1 meter per plant. The officials will decide whether a plant is damaged or not.

The task completing teams will be ranked by the results of resulting total distance values. The best 3 teams will be rewarded. This task 1, together with tasks 2 and 3, contributes to the overall contest winner 2014. Points for the overall winner will be given as described under chapter 0.2 Awards.

2. Task “Advanced navigation” (2)

2.1. Description

Under real field conditions crop plant growth is not uniform and even obstacles may occur. Furthermore, sometimes the crop rows are not even parallel. We will approach these field conditions in the second task.

The robots shall achieve as much distance as possible within 5 minutes while navigating between straight rows of maize plants, but the robots have to follow a certain predefined path pattern across the field (*picture 2* at the end of this text). Additionally at some locations plants will be missing (gaps) at either one or both sides with a maximum length of 1 meter. There will be no gaps at row entries.

In order to challenge the robots' abilities to navigate 2 obstacles - e.g. traffic cones - will be placed at not published positions between some rows and will block the path for the robot. The robot has to reverse and to continue in the described path pattern. The coded pattern takes blocked paths into account.

A headland width of not more than 1.5 meters will be available for turning.

The code of the path pattern through the maize field is done as follows: S means START, L means LEFT hand turn, R means RIGHT hand turn and F means FINISH. The number before the L or R represents the row that has to be entered after the turn and the single number 0 means return in the same path. Therefore, 2L means: Enter the second row after a left-hand turn, 3R means: Enter the third row after a right hand turn. The code for a path pattern for example may be given as: S - 3L - 0 - 2L - 2R - 1R - 5L - F.

The code of the path pattern is made available to the competitors 15 minutes before putting all robots into the parc fermé. Therefore, the teams will not get the opportunity to test it in the contest field.

2.2. Assessment

The distance travelled in 5 minutes is measured. If the end of the field is reached in less time, this actually used time will be used to calculate a

bonus factor = total distance * 5 minutes / measured time.

The total distance includes travelled distance and the penalty values. Distance and time are measured by the jury officials.

Manual intervention during the within field operation will result in a distance penalty of 3 meter per touch. During headland turning – after exiting the rows - to help the robot finding the right row will be punished with a penalty of 5 meters. The number of interventions (touches) will be counted by the officials.

Crop plant damage by the robot (e.g. bended, broken or uprooted plants) will result in a penalty of 1 meter per plant. The officials will decide whether a plant is damaged or not.

The task completing teams will be ranked by the results of resulting total distance values. The best 3 teams will be rewarded. This task 1, together with tasks 2 and 3, contributes to the overall contest winner 2014. Points for the overall winner will be given as described under chapter 0.2 Awards.

The *picture 2* shows an example of how the crop rows and the path tracks could look like for task 2. Be aware, the row gaps and the path pattern will be different during the contest!

3. Task "Professional Application" (3)

3.1. Description

The third task is based on a realistic scenario within precision farming. Five weed plants will be randomly placed within crop rows. These will be yellow golf balls placed on tees on the ground in line with the crop plants on the soil surface. During the run the weeds have to be indicated and mapped. By using an RTK GNSS system an absolute geo-referenced weed map has to be generated. A suitable

device[2] as a battery powered receiver with antenna and interface cable will be provided by the organiser. The specifications will be published in advance (size, weight, interface and data protocol). A testing box will be available for testing purposes the day before the contest. The submitted final map must consist of coordinates of the individual weed plants. The robot has 5 minutes to complete the run.

It will be a combined task consisting of three robot performance skills that need to be performed simultaneously during the run.

Subtask 1: Autonomous navigation between curved crop rows of maize plants each second row (!), not adjacent rows!

Subtask 2: The weed plants have to be indicated to the jury by very clear optical, acoustical or other signals while the machine is passing the weed.

Subtask 3: The weed plants have to be mapped with absolute coordinates by using the GNSS system. Immediately after the run the team has to deliver a text file[3] consisting of the values of the five coordinate pairs.

3.2. Assessment

For this task the robot shall navigate autonomously, but manual correction of the navigation performance is allowed but should be avoided. The total travelled distance will not be assessed.

Crop plant damage by the robot (e.g. bended, broken or uprooted plants) will result in a penalty of 0.25 point per plant. The officials will decide whether a plant is damaged or not.

The number of correctly indicated weed plants will be counted by the jury and points will be given for each correctly indicated weed (max. 5 points). The reference point on the machine must be visible e.g. by an indicator. Each wrongly indicated weed will be punished by 0.25 point value, but for the total sum the lowest value is zero.

The generated map consisting of coordinates (x, y values in meters) of the weed plants will be analysed. If the error (distance between true and mapped weed coordinate) is less than 0.75 meter a point will be given as correctly mapped weed (max. 5 points). If teams have the same number of points then the mean error of all coordinates will be used for the ranking (the smaller the better). Files with more than 5 coordinate pairs will not be assessed. After the run the text file must be delivered to the parc fermé officer e.g. file saved on a USB stick.

Before the run of each team the GNSS-box will be checked concerning the RTK fix status[4].

The task completing teams will be ranked by the number of points for correctly indicated weeds (max. 5) and correctly mapping weeds (max. 5 and perhaps mean error). The best 3 teams will be rewarded. This task 3, together with tasks 1 and 2,

[2] Coordinates will be in UTM (NMEA $PTNL, PJK string), output frequency 5 Hz

[3] Including team name, date and time stamp, data pairs of number of detected weed and coordinates (easting and northing in meters with 3 decimal points). There shall be no further information in the text files. An example file will be on the FRE 2014 webpage on the download flag.

[4] The robot is welcome to also indicate GNSS mode status.

contributes to the overall contest winner 2014. Points for the overall winner will be given as described in chapter 0.2 Awards.

4. Task "Cooperation" (4)

4.1. Description

Two-team groups will conduct a cooperative task. The groups are free to define their tasks as long as it is a task with two robots working together. For this purpose there has to be a somehow communication between the robots. However, the robots could also “communicate” via pressure sensors or vision etc. Everything is possible in this task as long as it is cooperative. The communication could also be done by Wi-Fi and / or ISO 11783 protocol. Nevertheless every other way of communication is allowed and we are open for good ideas. This is a nice step forward in technology because communication between field robots will be very important in the future.

In 2014 we are allowing to use the 2 available GNSS systems. Therefore, two collaborating machines can base their performance on absolute positioning. The organisers must be informed in advance if teams want to go for this option.

The teams have to indicate their participation during the contest registration. For the contest they will be chosen by the organizer and will be pronounced as early as possible. Team groups will have a time limit of five minutes for conductance.

4.2. Assessment

The jury will assess the (i) underlying idea, the (ii) technical challenge and the (iii) robot performances by giving points from 1 (fair) to 10 (excellent) for each. The three criteria will be weighted by factors 1, 1 and 2. The teams will be ranked by highest points.

The task 4 is optional and will be awarded separately. It will not contribute to the contest winner 2014.

5. Task “Freestyle” (5)

5.1. Description

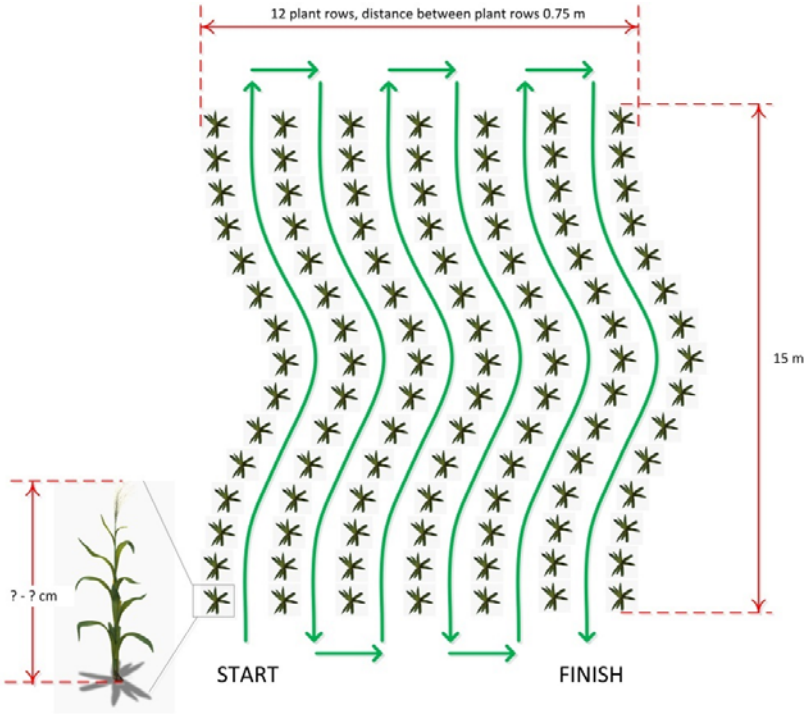
Teams are invited to let their robots perform a freestyle operation. Creativity and fun is required for this task as well as an application-oriented performance. One team member has to present the idea, the realization and perhaps to comment the robot’s performance to the jury and the audience. The freestyle task should be related to an agricultural application. Teams will have a time limit of five minutes for the presentation including the robot’s performance.

5.2. Assessment

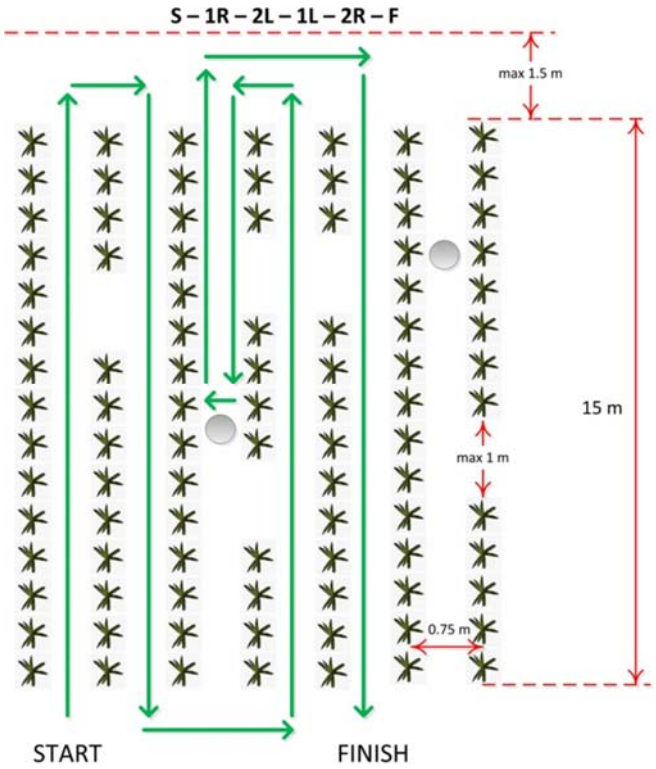
The jury will assess the (i) underlying idea, the (ii) technical challenge and the (iii) robot performance by giving points from 1 (fair) to 10 (excellent) for each. The three criteria will be weighted by factors 1, 1 and 2. The teams will be ranked by highest points.

The task 5 is optional and will be awarded separately. It will not contribute to the contest winner 2014.

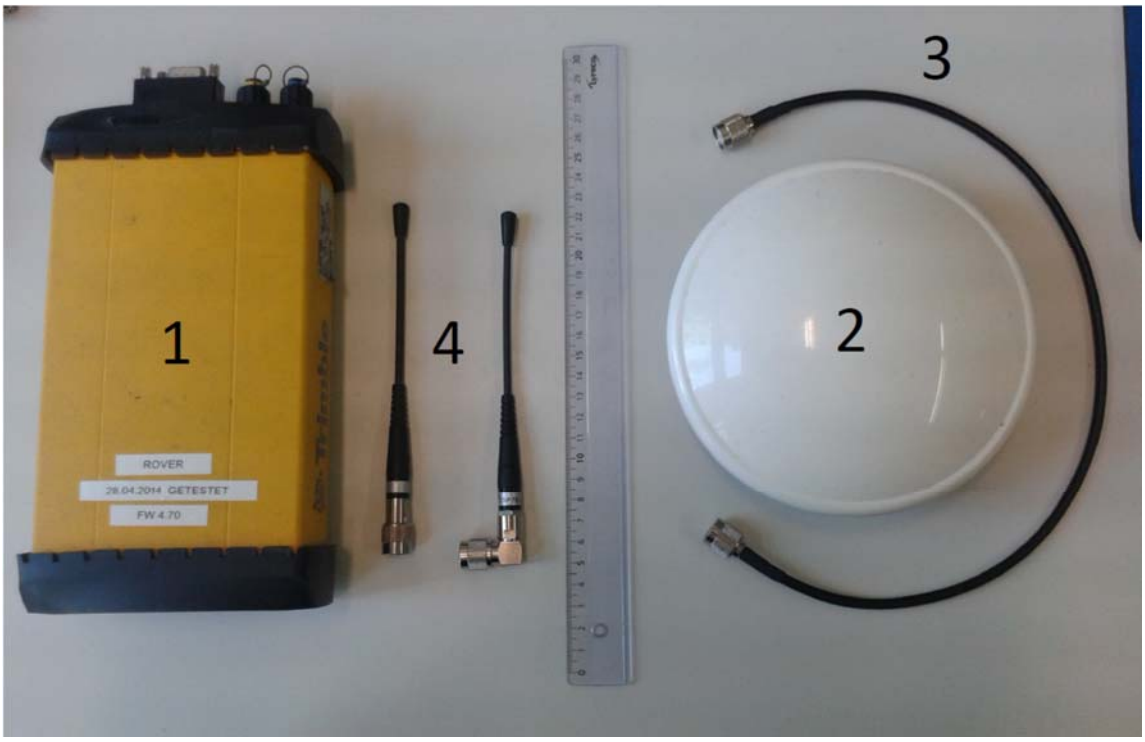
Appendix



Picture 1 – Dimensions and row pattern for task 1



Picture 2 – Dimensions and row pattern for task 2



Picture 3 – GNSS system (Trimble AgGPS RTK Base 450): (1) receiver, (2) antenna Trimble AG25 GNSS with magnetic base, (3) cable for connecting the satellite antenna to the receiver and two options regarding the (4) radio antenna for RTK correction signal. No power supply is required due to built-in battery.

3. Robot information

AGROB V14 – Towards a low cost vineyard robot

Filipe Neves dos Santos[1,2], Márcio Monterroso[3], Raul Morais[1,3]

[1] *INESC TEC - INESC Technology and Science, Porto, Portugal*
[2] *FEUP - Faculty of Engineering, University of Porto, Portugal*
[3] *UTAD – Universidade de Trás-os-Montes e Alto Douro, Portugal*

1. Introduction

AGROB V14 [9] is the name of our first, small and low cost outdoor robot, for vineyard monitoring, which will be tested on the field robot competition 2014.

This robot is built on top of a commercial radio-controlled model (RC-model) where are attached two tiny computers, with Advanced RISC Machine (ARM) processors, running Linux and the robotic operating system (ROS). A laser Range Finder (LRF), distance infrared (IR) sensors, two low cost cameras, inertial measurement unit and GPS receiver are the main information sources available to the robot for the task execution.

The main goal of this work is to build a low cost and robust robot that could be used on the vineyard monitoring tasks, such as vegetation detection [1], leaf area index [2], and yield estimation [3][4].

2. Mechanics

AGROB V14 is built on top of RC model, Traxxas E-Maxx (figure 1), a 1/10 scale 4WD Electric Monster Truck [5].

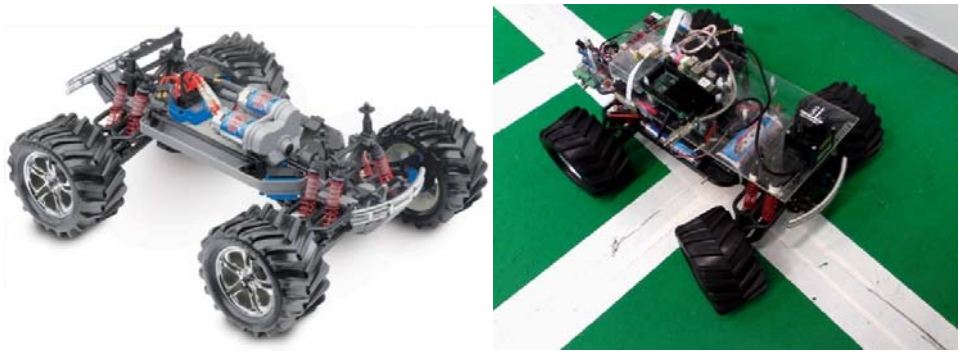


Figure 1. On the left the original RC model. On the right the RC model with sensors and processing units.

3. Sensors and Software

3.1 Hardware

A laser Range Finder (LRF), distance infrared (IR) sensors, two low cost cameras, inertial measurement unit and GPS receiver are the main information sources available to the robot for the task execution. The main specifications of these sensors are shown on table 1.

Sensor	Maker	Model	Main Specifications
LRF	Hokuyo	URG-04LX-UG01	Detection distance 20mm ~ 4000mm , Accuracy Distance 20mm ~ 1000mm : ±30mm, Distance 20mm ~ 4000mm : ±3% of measurement, Resolution 1 mm, Scan Angle 240°, Angular Resolution 0.36°, Scan Time 100msec/scan. [6]
IMU	Sparkfun	9DOF Razor IMU	3 accelerometers, 3 Magnetometers, 3 Gyros. [7]
Distance IR	Sharp	GP2Y0a21YK0F	Detection distance: 100mm to 800mm [8]
Camera IR	RaspberryPi	IR	5MP CMOS, photos 2592 x 1944, video: 1080p at 30 FPS, 720p at 60 FPS and 640x480 at 60 at 90 FPS.
Camera RGB	UDOO	RGB	5MP CMOS, photos 2592 x 1944, video: 1080p at 30 FPS, 720p at 60 FPS and 640x480 at 60 at 90 FPS.

Table 1. Agrob V14 - Sensor Specifications

The sensor information acquisition and processing, and Agrob V14 control is executed by two tiny computers.

Board	CPU/GPU	Memory	SD card/OS
UDOO	Freescale I.MX 6 ARM Cortex-A9 Quad Core 1 GHz/ GPU Vivante GC 2000 + Vivante GC355+Vivante GC320	1GB	16 GB/ Linux, Ubuntu 13.04, ROS hydro
RaspberryPi	ARM1176JZF-S @ 700 MHz/ GPU VideoCore IV	512MB	8 GB/ Linux, Ubuntu 13.04, ROS hydro

Table 2. Agrob V14 – Processing units

The motor controller used is the EVX Model 3014, which is connected to the RaspberryPi through the Pololu SSC03A board, as shown on figure 2.

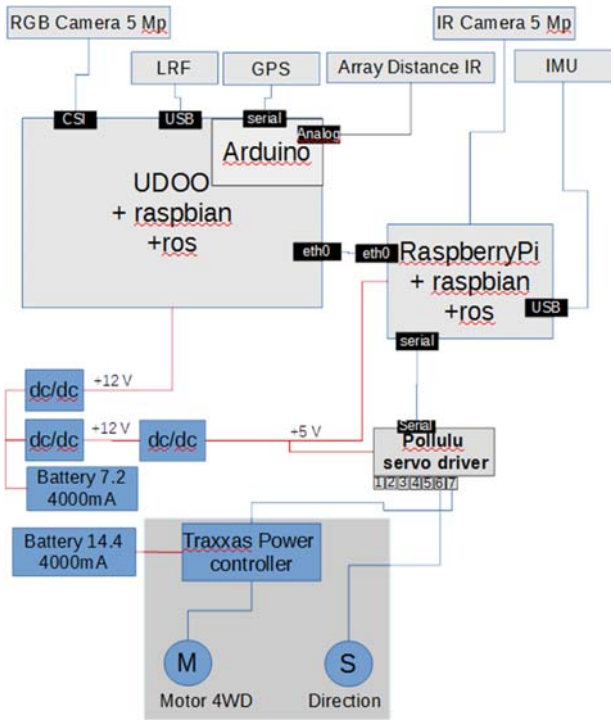


Figure 2. Agrob V14 general connections diagram.

3.2 Software and strategy

This robot has two processing units running Linux (Ubuntu 12.04) with the robotic operating system (ROS-groovy) on top.

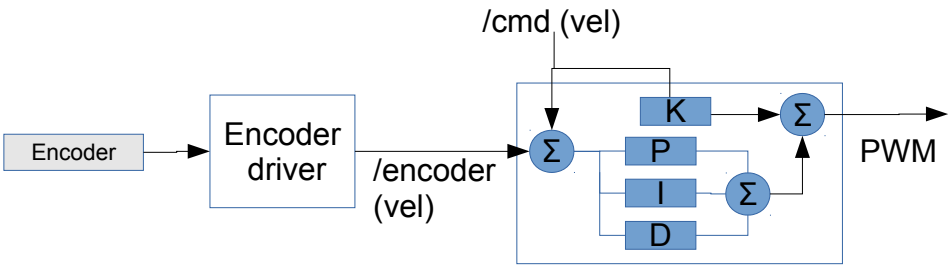


Figure 3. Velocity controller.

In order to accurate control the robot velocity, a proportional integral derivative controller (PID) with a feedforward controller is used. The encoder signal is processed on the arduino module of UDOO board, where is published the robot velocity through the encoder topic message. This velocity information feeds the PID and feedforward controller, as shown on figure 3.

Besides the sensors and actuators drivers, there are five main nodes, running on ROS: Task executer, Maneuverer, Collision avoidance, Localisation and Mapping, and the visual plants detection, as shown in figure 4.

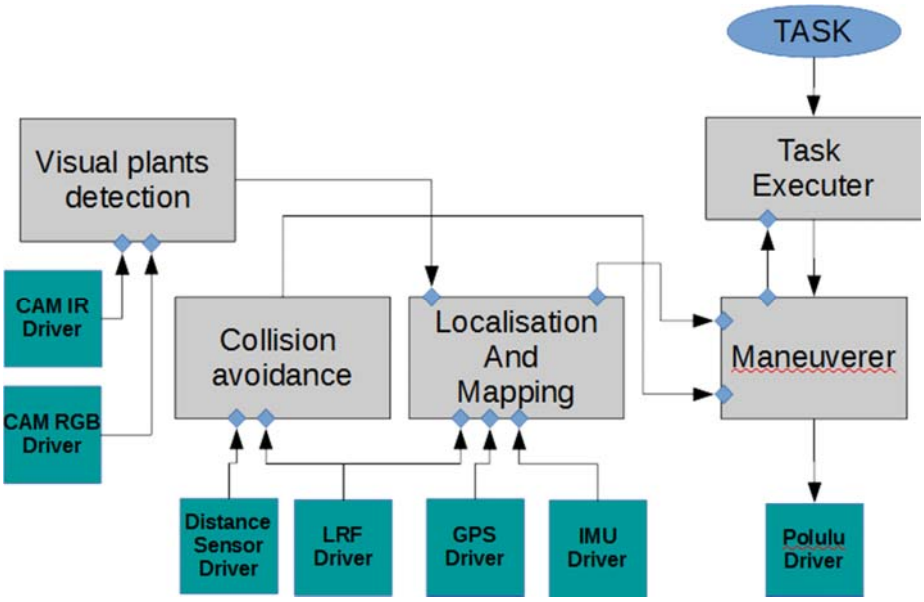


Figure 4. Agrob V14 Main ROS nodes

In order to make possible the collision avoidance and navigation a local occupation grid map is generated in *Localisation and Mapping* module and then published. The grid

map is defined by an integer matrix, where each cell stores the existence or inexistence of obstacles. This occupation grid map is updated by using the laser range finder observations and by the robot motion estimation.

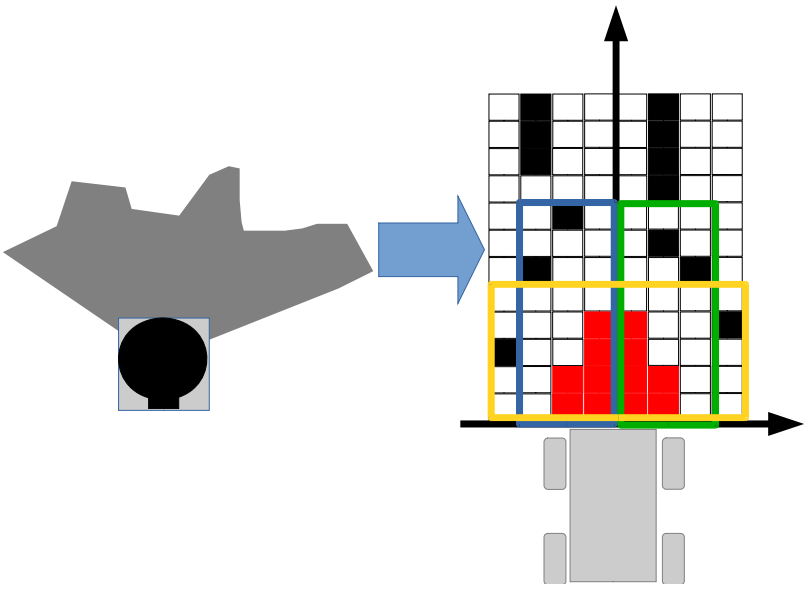


Figure 5. Local occupation grid map is generated in *Localisation and Mapping* module. On the left LRF observation, on the right the updated grid map.

The *collision avoidance* module defines four main regions on the local map, figure 5. The red region defines a region where the robot cannot avoid a collision, due the maximum turning rate of the robot. The yellow region is used to detect the end of weeds line, when this region is empty is published a topic notifying the end of line, this information is used the task executer. The blue and green regions are used to estimate the turning rate of the robot in order to move safely through objects, this turning rate is obtained in each occupied cell:

$$w = \sum_{n=0}^{N} w_n \quad \textbf{\textit{Eq.1}}$$

Where:

$$w_n = k.\tan^{-1}\left(\frac{x_p}{y_p}\right) \quad \textbf{\textit{Eq.2}}$$

Where, *k* is the gain parameter, and x_p, y_p are the coordinates of the occupied cell.

The *task executer* was been built with a four state machine and the *maneuverer* module was built with a seven state machine, as shown on figure 6.

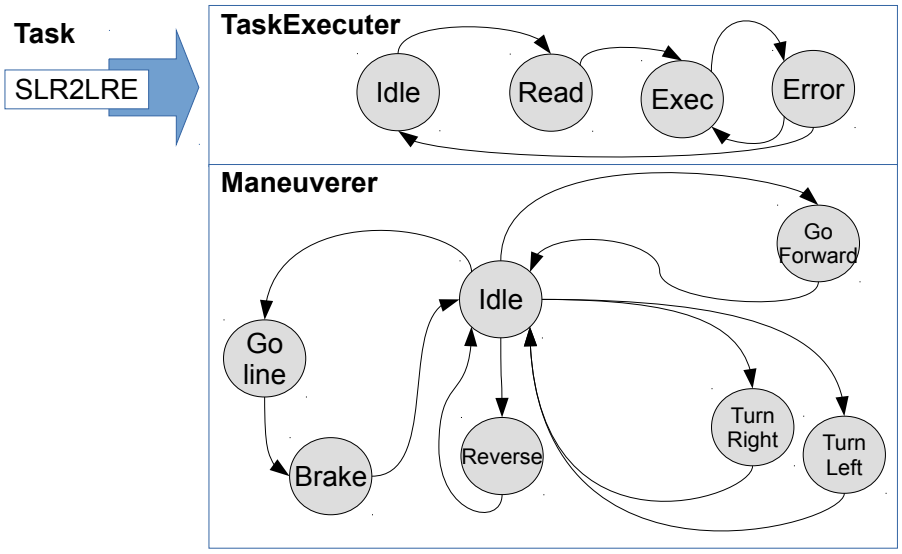


Figure 6. Task executer and maneuverer modules

The robot accepts tasks which are defined by a string using the fieldrobot standard. Each character defines the sequential maneuver that should be executed by the robot. This character can take a number or (***S,L,R,E***), where ***S*** defines the robot start operation and makes the robot move through the weed line until reach the line end, ***nL*** defines that the robot should turn left and align to the **n** row ($n \in \aleph$), ***nR*** defines that the robot should turn right and align to the **n** row ($n \in \aleph$) – by default ***n***=1, and **E** defines the robot stop operation.

Due the low turning radius of the robot, the turning maneuvers are decomposed into five simpler maneuvers, as shown in figure 7.

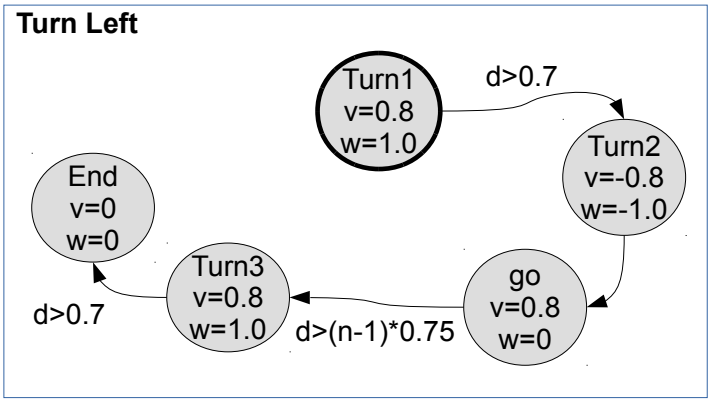


Figure 7. Turn left maneuverer

The turning maneuver is configured by a single parameter, the number of next row to follow.

4. Conclusion

During this work, the robot agrob V14 was been built, tested and benchmarked on tasks 1 and 2 of fieldrobot 2014 event.

On task 1, this robot has been ranked on the 12 position (in 23 teams). The robot was able to move autonomously through two rows, 15 meters each, and execute the complex turning maneuverer, in less than three minutes [9].

One disadvantage of this robot is the low turning radius, which requires a complex maneuverer for the row transition. Other disadvantages are the missing of: odometry based on laser range finder observations, local map updates through the visual installed cameras, and redundant sensors.

The advantage of this robot is the use of simple and modular software architecture, over ROS, which can be extended to perform complex tasks. Another advantage is the use of robust and low cost hardware which makes this robot easily replicated at low cost, less than 3000€.

As a future work we intend to add redundant sensors (such as backward laser range finder), increase the local map accuracy in order to make possible to increase the maximum robot velocity, and add a SLAM approach able to work on the fieldrobot event scenario.

The source code and results of this work can be found at http://hyselam.com/agrob.html.

5. References

[1] Bradley, D., Thayer, S., Stentz, A., & Rander, P. (2004). Vegetation detection for mobile robot navigation. *Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-04-12*.

[2] Leemans, V., Dumont, B., Destain, M. F., Vancutsem, F., & Bodson, B. (2012). A method for plant leaf area measurement by using stereo vision. In*Proceedings of CIGR-AgEng 2012 International Conference on Agricultural Engineering*.

[3] Nuske, S., Achar, S., Bates, T., Narasimhan, S., & Singh, S. (2011, September). Yield estimation in vineyards by visual grape detection. In*Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on* (pp. 2352-2358). IEEE.

[4] Reis, Manuel JCS, et al. "Automatic detection of bunches of grapes in natural environment from color images." *Journal of Applied Logic* 10.4 (2012): 285-290.

[5] Traxxas E-Maxx: Technical specifications. Available at: http://traxxas.com/products/models/electric/3903emaxx-downloads (26-May-2014)

[6] LRF Hokuyo URG-04LX-UG01 specifications. Available at: http://www.hokuyo-aut.jp/02sensor/07scanner/download/products/urg-04lx-ug01/data/URG-04LX_UG01_spec_en.pdf (26-May-2014)

[7] Sparkfun IMU-9dof specifications. Available at: https://www.sparkfun.com/products/10736 (26-May-2014)

[8] Sharp GP2Y0A21YK0F specifications. Available at: http://www.sharpsma.com/webfm_send/1489 (26-May-2014)

[9] AGROB V14 webpage, Available at: http://hyselam.com/agrob.html (26-May-2014)

Agro-Bot

HAMK
UNIVERSITY OF APPLIED SCIENCES

Samrat Gautam, To Doan Ngoc Hai, Tran Phu Long and Andrei Sandru, Markku Kippola

University of Applied Sciences, Valkeakoski, Finland

1. Introduction

The aim of this project was to design a field robot and to participate in the Field Robot Event 2014 “Robot Fun and Creativity” (later FRE 2014). FRE 2014 is unique from other types of robot competitions because a proto type model must be built with an ability to perform its tasks in a real world environment. This was the most challenging task as a proto type model works well in a simulated or in a controlled environment, whereas in an outdoor environment and especially in an agricultural field most of the parameters change with respect to time. It is impossible to track or predict the entire trend by using any known mathematical models. Because of this designing a prototype robot with a higher degree of stability, robustness and disturbance rejection feature is still a huge challenge for engineers and designers. Hence, several theories, research papers and previous FRE videos were carefully studied for better tracking and sensing such trends. To precisely control the robot, to achieve better and quicker response to a changing environment, different controllers were designed and switched accordingly for examples: navigation, the turning mode and the reverse drive.

A team of five members worked together in this project to take part in FRE 2014. Tasks were divided among members. Task one: designing agro-robot, task two: designing follower robot for irrigation, task three: navigation based on an image captured from the camera, task four: navigation based on RTK GNSS, task five: navigation based on distance measurement from sonar sensor and task six: wireless communication between two robots. Before starting to fabricate the robot several sensors were tested and analysed such as; an IR sharp Sensor, a sonar sensor, a wheel encoder, a digital gyroscope and a camera. As the robot had to drive on loose muddy terrain four-wheel differential drives with a skid-steered chassis (each wheel with a separate motor drive) was fabricated. The arena of FRE was in an outdoor environment so there was a possibility to fluctuate the disturbances such as; intensity of sun light; temperature and humidity. This fluctuation could affect the stability and robustness so the best two different algorithms were used for the final event. This was verified by conducting several tests in a simulated maize field build of paper and plastic indoors and by adding possible disturbances.

Out of the different sensors the vision based sensor was found to be the best and cheapest to sense a maize row when the lightning condition was good enough.

However, the data acquired from the vision sensor contained a lot of noise and was relatively large. Processing such a large amount of data was quite challenging and required a better controller. So a computer was used to process the data acquired from the vision sensor and two Arduino Mega microcontrollers were used to interface other sensors and actuators.

2. Mechanics

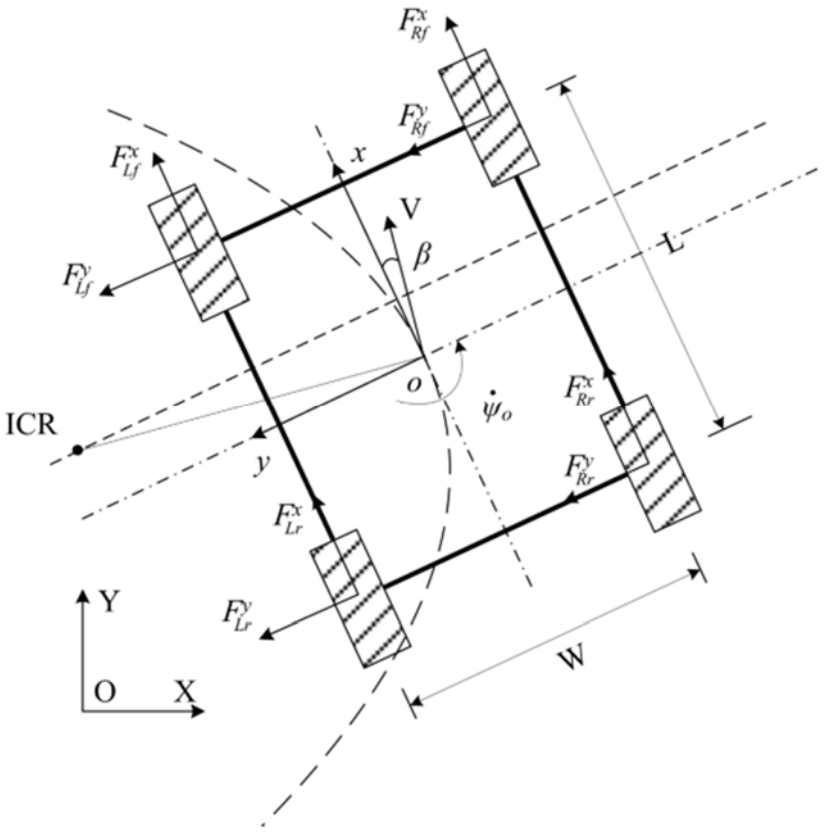
Research paper on differential speed steering control for four wheel independent drive electric vehicles by was preferred to be examine because the dynamics of our model were best described on it. As shown in figure, in case of skid steering four wheels were fixed. Hence, to turn the robot in a desired direction a suitable lateral slip must be developed in the inner wheels. Control strategies for lateral slip could be achieved in three different ways.

Control strategy one: in this control mode, with only the outer wheels the velocity is increased without changing the inner wheels with velocity. This was easy to carry out but the radius of turn was found to be bigger. It was because the velocity of the inner wheels may be at a higher value (Wu et al. 2013).

Control strategy two: in this control mode, the turning was obtained by locking the rotor of the inner wheels and introducing suitable velocity on the outer wheels (Wu et al. 2013). This type of control mode was suitable when a small radius of turning was required. However, it had one disadvantage: for a jerk free motion the robot had to stop before steering.

Control strategy three: in this control mode the steering radius was achieved by introducing differential speed in the inner and outer wheels (Wu et al. 2013). Provision of individual wheel velocity control of the inner and outer wheels had added additional flexibility to the control steering motion. The robot needed not to be stopped for steering a sharp turn.

As suggested by Wu, Xu and Wang (2013) and an experiment carried out by the authors in a controlled environment the third control strategy was used: increasing velocity of the outer wheels and reducing the velocity of the inner wheels as desired. However, in our case the point of contact was loose soil and predicting precisely the radius of turning by a controller was quite challenging. This was because lateral slip also depends upon the coefficient of friction developed between the wheel and the floor contact point.



Skid-Steered Vehicle (Wu et al. 2013.)

A kinematic model of a differential-steered vehicle moving at constant velocity about an instantaneous centre of rotation (later ICR) is shown in figure . In this model the following assumption was made to simplify the mathematical modelling: the centre of mass was at the geometric centre of the robot chassis, the robot runs at slow speed and two motors on the same side were provided with an electrical power producing the same speed. The kinematic relationship between the whole vehicle and the wheel speed can be expressed by a mathematical formula, which is shown in equation 1 (Wu, et al. 2013.)

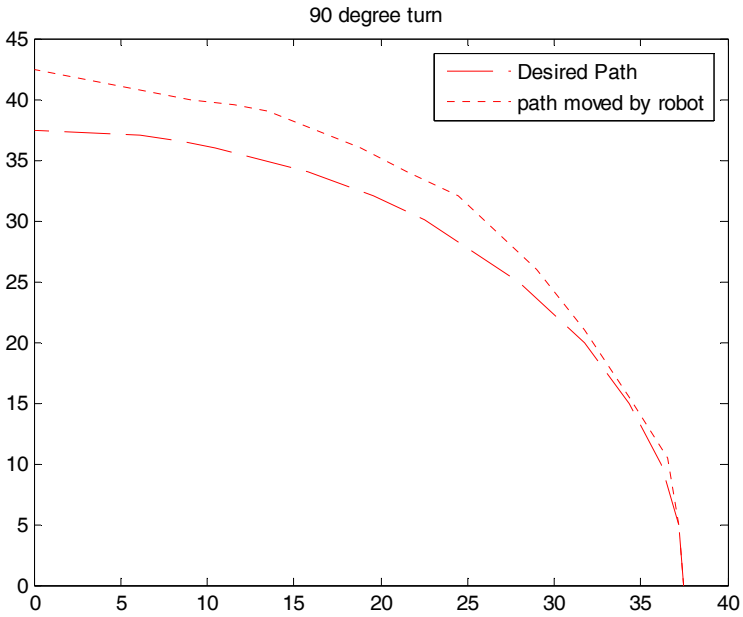
$$\begin{pmatrix} Vox \\ Voy \\ \psi o \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{ICRx}{W} & -\frac{ICRx}{W} \\ \frac{1}{W} & \frac{1}{W} \end{pmatrix} \begin{pmatrix} Vl \\ Vr \end{pmatrix} \qquad (1)$$

The W is the width of the robot

Vl is the left wheel velocity

Vr is the right wheel velocity

ICR is the instantaneous centre of rotation



Graph of the robot90 degree turn. (Gautam 2014)

A suitable algorithm was developed to execute the formula derived from Wu, et al. (2013) model. As shown in Fig. , the robot's path was deviated by a small amount. This was because of poor wheel coupling as there was a slip between the wheel and the shaft. Though the same power was supplied to all the wheels, yet there was some difference in speed. This leads to a deviation of the robot's path. Despite mechanical inefficiency, the results in the graph demonstrated that the differential speed steering control for a four wheel independent drive electric vehicle model best suited this project.

Suspension System

After conducting several tests with a 4WD chassis it was found out that the wheel lost contact if the terrain was uneven or loosely packed. In order to overcome this issue, suitable suspension was developed and implemented. In order to reduce jerk produced by a high torque motor and to establish a flexible connection between the wheel and the chassis, each wheel was fitted with suitable suspension system.

Suspension system for Skid-Steered Vehicle (Gautam 2014)

As shows in figure 7 a helical spring was used with a central rod. This rod was used to prevent the spring from a buckling effect. The total mass of the robot was 12kg. Approximately 20 kg of counter mass was required to compress the spring fully.

2.1 Agro-Bot

Agro-Bot was the name given to the robot by the authors. In this chapter the authors will discuss general aspects of the robot. The robot chassis was designed in Autodesk inventor 2012 and fabricated using aluminium AW 6063 T5 rectangular bar of size (100x20x2, 40x20x1.5 and 20x15x1.5) mm. An image of CAD designed Agro-Bot is attached as Appendix 3. Each wheel was separately powered by an electric motor and featured with a suspension system to ensure that all the wheels touched the ground. The robot chassis held everything attached to it and had a provision to hook follower.

Agro-bot (Gautam 2014)

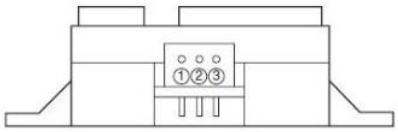
As shown in figure 8, it had one magnetic compass which was denoted by 'M', it had four sonar sensors whose positions were defined as -90, -45, 0, 45, 90. It had a computer on the top which controls outputs through Arduino. 12V 15Ah battery was placed just below laptop. Motor H-bridge, power adapter and Arduino were placed inside a junction box which is underneath a laptop. There were two cameras; one in the middle and the other on right hand side. Rare wheels were equipped with a wheel encoder. A camera placed on the centre was used to capture images for detecting a yellow coloured ball. The camera placed on the right hand side was used to synchronize the robot path.

3. Sensors

The selection was based on the test results and costs. The first test was carried out with an infra red sharp sensor and it was found out that this type of sensor works well only in normal lightning or in an absence of ambient light. For the second test a Sonar Sensor was coupled on a Servo motor to mimic a laser range scanner. This was because a laser range scanner was very expensive. The results from this design were good but for a single scan of 180 degrees with a resolution of 5 degrees per step it took more than 5 seconds. With this delay value it was impossible to complete even the first task in FRE. So we had only one option remaining which was to make a skirt of a sonar sensor on the chassis. For this purpose five sensors were placed at an angle of -90, -45, 0, 45 and 90. The main problem with a sonar sensor was that soft materials absorb considerable amount of signals because of which, the receiver failed to detect this echo.

IR Sharp Sensors

An infra red (later IR) sharp sensor for Arduino is a commercial sensor produced for educational projects. It consists of an infra red emitter and a photo diode which are separated by a small distance. IR rays projected by an emitter will be reflected back to the receiver if an object is introduced within its range. It is moderately accurate for measuring short distances under normal ambient light. We cannot see IR rays by our eyes but we can use a camera to verify the presence of light. The IR-sharp sensor used for the test purpose is shown in figure 9 (a) and the pin configuration is shown in figure 9 (b).



PIN	SIGNAL NAME
①	V_O
②	GND
③	V_{CC}

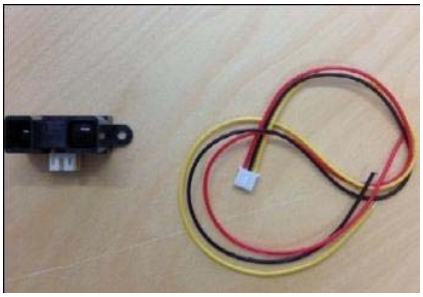
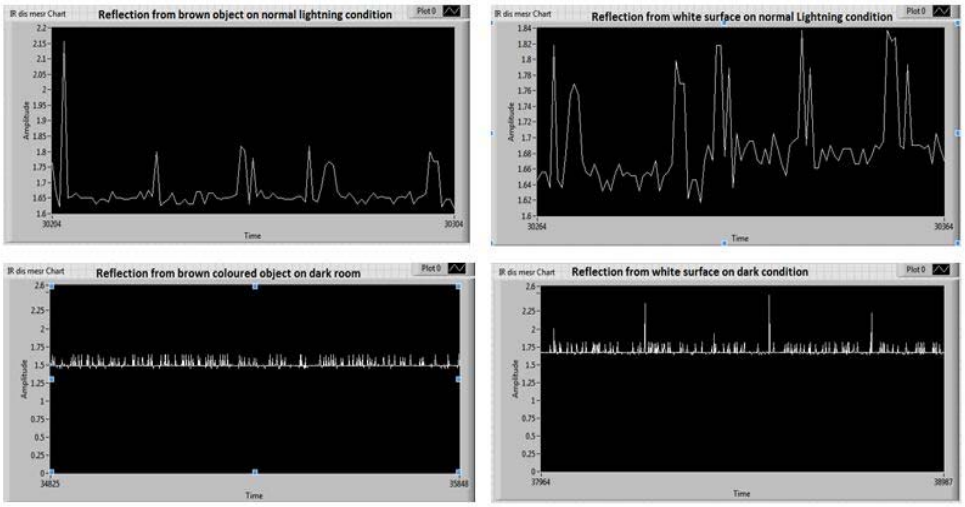


Figure (a) Figure (b)

IR sharp sensor (Gautam 2014)



IR sharp sensor disturbance measurement in real time (Gautam 2014)

An experiment carried out in four different conditions is shown in figure 10. The amplitude of the signal was plotted on the x-axis and time on the Y-axis. The amplitude of the signal ranges from 1.6-2.2 volts. The top left graph shows the IR signal received from a brown surface in normal lightning conditions. The signal was affected by noise. The top right graph shows the same signal from a white surface in normal lightning condition. The signal thus obtained was highly affected by noise. The bottom left figure shows the same signal from a brown surface but without any light condition. This signal was quite good. The bottom right corner shows the same signal on a white surface and in no light conditions. Hence, it was clear from the graphs that IR sensors were mostly affected by intensity of light. So as an alternative to this sensor, a sonar sensor was chosen.

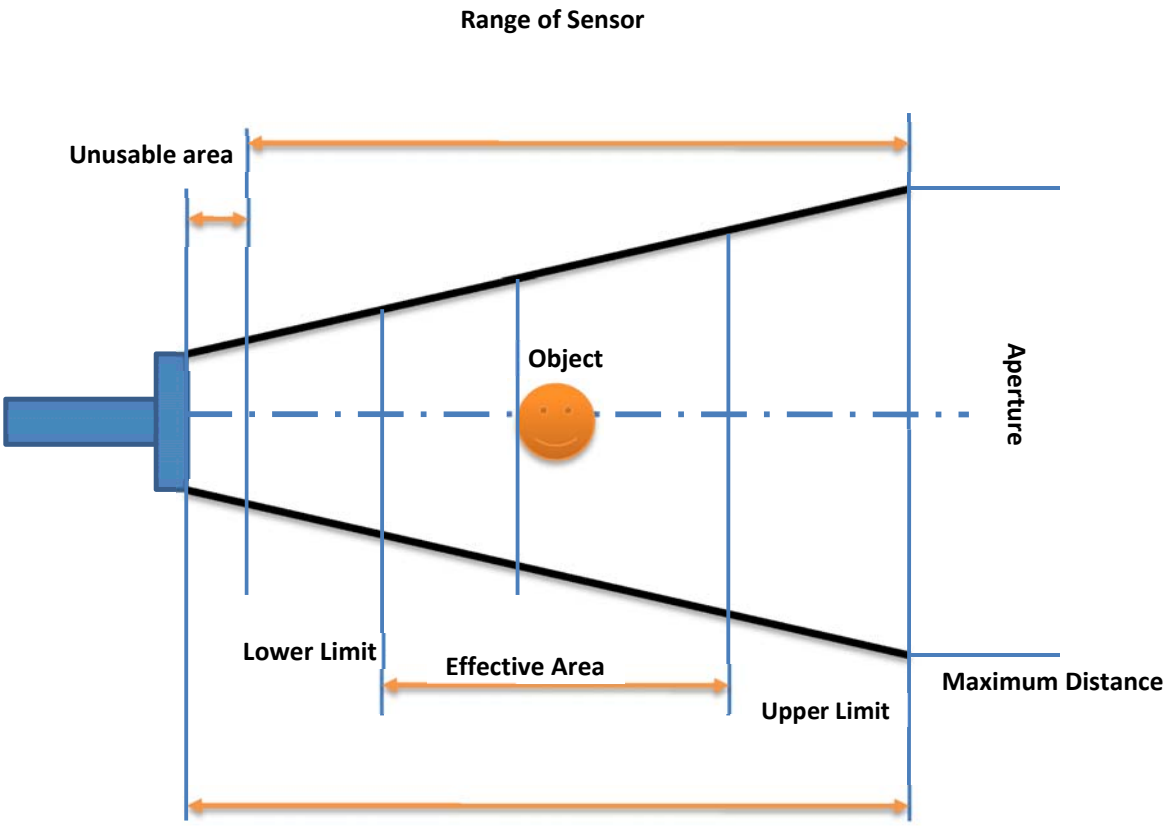
Ultrasonic Sonar Sensor

Ultrasonic sonar sensor is a type of active sensor which has a transducer and a receiver. When an electric signal is supplied to a transducer made of thin membrane of piezoelectric ceramic then mechanical distortion occurs; emitting ultrasonic waves. These waves travel through air and produce an echo if a foreign body is introduced within it range. The reflected waves (echo) were picked by receiver producing mechanical vibration on a thin membrane of piezoelectric ceramic creating an electrical signal. Ultrasonic wave has sorter wave length which means better accuracy can be achieved on distance measurement. The speed of ultrasonic waves mainly depend on temperature, but the amplitude of the echo waves depend on the area and surface properties of reflecting materials (Foessel 2000).

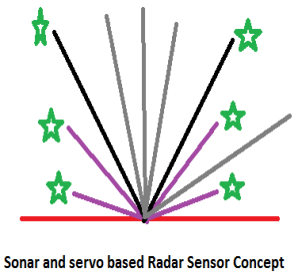
Sonar sensor and servo module (Gautam 2014)

First of all an experiment was carried out with a sonar sensor and a servo motor (as shown in figure11) to mimic laser range scanner. In this experiment a sonar sensor of range between 0.04-4m was mounted on a shaft of servo motor. The resolution of 5 degree per step and a scan angle of 180 degree were chosen. A trigger was send to transmitter at every step and distance corresponding to eco was stored in 2D array before taking new step. The outcomes of this type of combination were found to be satisfactory if the object size was relatively bigger, sampling time was longer and scan area was small.

But in this project stem of maize plant was no more than 0.04m, the distance between two successive plants was 0.133 m and distance between two rows was 0.75m (FRE 2014) and sonar sensor used in this event had an aperture angle ranging from 30-45 degree as shown in figure 12. Two or more than two plant can be enclosed within effective region. And another major drawback was long sampling time.



Sonar sensor effective area (Foessel 2000)



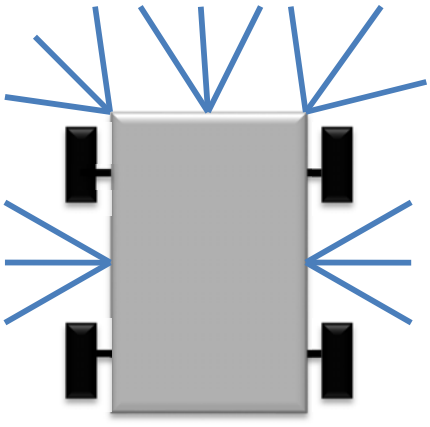


Figure (a) Figure (b)

Sonar sensor position in the robotchassis (Gautam 2014)

For a single scan of 180 degree (as shown in figure 13 (a)) with a resolution of 5 degree per step size; sampling time was more than 3 second. Hence, array of 5 sensors (as shown in figure13 (b)) were used to acquire distance in order to reduce sampling time. Schematic of sonar and servo module is attached in appendix 3

Wheel Encoders

Wheel Encoders used in this robot was hand fabricated. It is an internal sensor which provides angular velocity of the motor. Equally distributed strip of black pattern is printed on transparent plastic and glued between two circular disks to strengthen it. Diameter of circular plastic template exceed by 6 mm. Sensor is made up of optical transmitter emitting very narrow beam of light and a receiver which is facing each other. Very narrow optical beam is blocked by this small and equally distributed pattern when shaft attached to the wheel is rotated. As a result square wave is produced at output terminal of the encoder. Square wave received from the encoder is enough to determine revolution per minute. However, it failed to provide direction of rotation. Therefore, two optical encoders were used at 90 degree phase shift to each other to determine direction of rotation (Olson, 2009). The schematic of wheel encoder with sensors is shown in figure 14 (a), determining direction from quadrature phase wheel encoder is shown in figure 14 (b) and the actual wheel encoder output is shown in figure 14 (c).

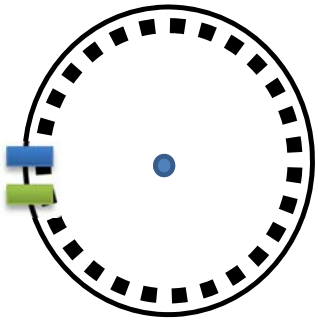


Figure (a): wheel encoder

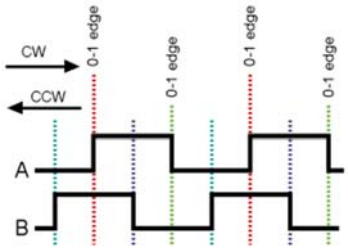


Figure (b): Direction signal

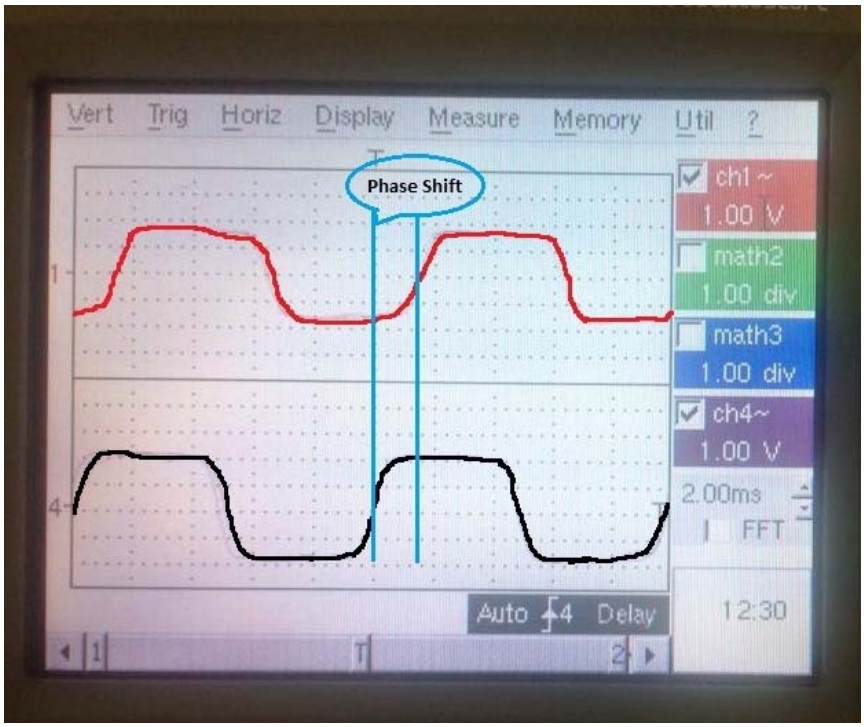


Figure (c)

Quadrature Phase wheel encoder (Gautam 2014)

$$Resolution = \frac{total\ pulse}{360^0} \qquad (2)$$

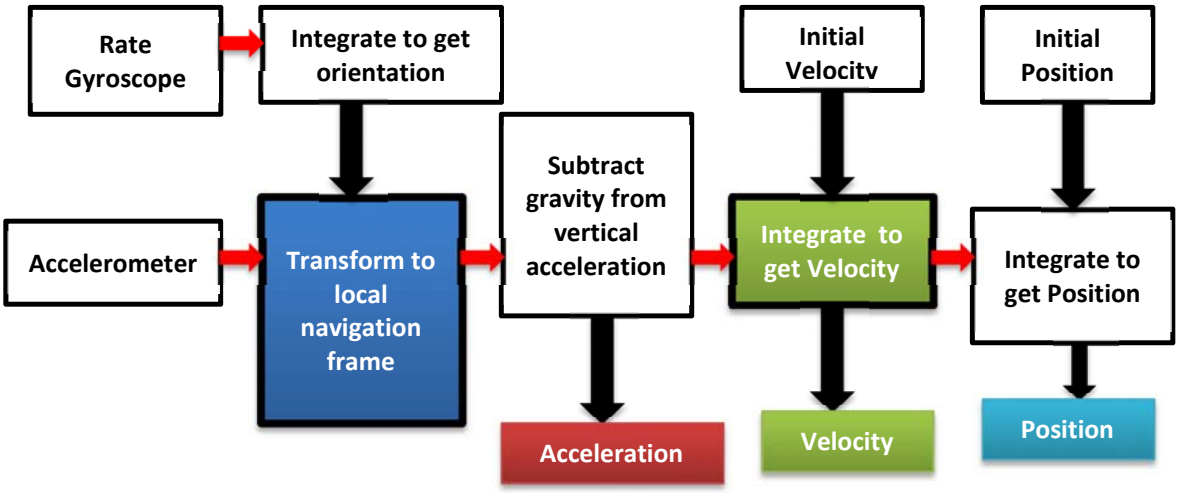
Based on the equation (2), in one rotation it produces 64 ticks, resolution is 0.177 per degree and smallest angle it can detect is 5.62 degree. Output signal of wheel encoder fabricated for this project deviates significantly from an ideal wheel encoder. Phase difference between two channels in an ideal wheel encoder is exactly 90 degree. Because of design constrains, the authors was able to achieve maximum of 55 degree phase shift. However, it was sufficient to distinguish clockwise and counter clockwise direction.

Inertial Measurement Unit

Inertial measurement unit (later IMU), consist of accelerometer, gyroscope and manometer. Accelerometer is used to measure linear acceleration, and gyroscope to measure rotational velocity. Figure 16 shows the integration process of these two sensors in a sequential block. Combination of this two sensing element on one package helps to sense motion type, rate, gravitational force and direction (Hazry, Sofian, Azfar 2009). It is used in real time navigation to calculate the acceleration, velocity and

position of moving system. This is obtained by integrating the acceleration and rotational rates signals from IMU.

Measurements of the direction of Earth's gravitational and magnetic field vectors along with the angular rates allow estimate of orientation of the sensors module. This sensor is mounted on the frame of robot. These orientations in turn are used to transform acceleration measurement from the moving body coordinate frame to an earth fixed reference frame. The total acceleration is subtracted from gravitational acceleration. The remaining acceleration is double integrated to estimate position relative to the starting point as shown in figure16. (Vincent 2013)



Linear acceleration and orientation (Gautam 2014)

IMU was used in our application because the resolution was good and data from wheel encoder was affected if any slip occurs on the wheel. It was also helpful to navigate in between crops rows as the array of data contains orientation about the reference, linear acceleration, and a rotational velocity of the robot. However, data from IMU was found to be affected by drift, accumulative error and sine error.

3.1 Hardware

Large amount of data needed to be computed when vision was used as primary source of data. Microcontroller such as Arduino cannot handle such a large amount of data; hence personal computer was used for extracting features from vision sensors. Computer running on windows 7 was used by disabling unwanted features of windows 7 for this task. Very few programs were operated on this computer. So that optimum performance was achieved to perform real time computation. Two Logitech USB cameras were directly connected to computer. National Instruments LabView 2010

was used to compute data acquired from two cameras. Sensors data was acquired from Arduino Mega 1 suitable control output was send to Arduino Mega 2 to energize the actuators'.

Laptop manufactured by Asus was used. The configuration of laptop are following:

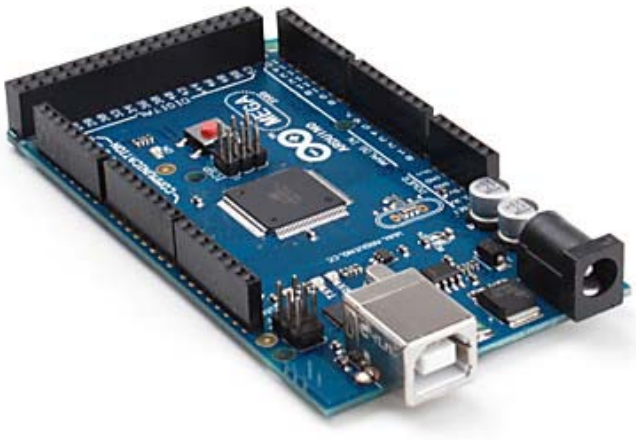
Processor: Intel(R) Core (TM) i5-2450M CPU @ 2.50GHz

RAM: 4.00 GB

Operating System: Windows 7, 64-bit Operating System

Interface Board

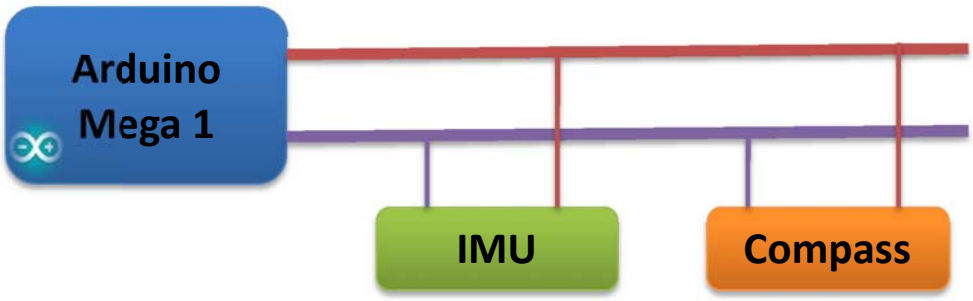
Computer could not directly communicate with sensors and actuators'. In order to establish basic communication between these devices an interface board was needed. Hence, to connect sensor and to control actuators, microcontroller was used. Two units of Arduino Mega microcontroller were used as an interface board to computer. First and for most reason to choose Arduino was that: it is open source, variety of sensors and other module was available in the market and next, it is reliable compared to self built microcontroller circuit. All the inputs were connected to Arduino Mega-1 and all the outputs were connected to Arduino Mega 2. However, in Arduino Mega-2 signals from wheel encoder was connected. It is because in Arduino Mega-2 PID controller were used to drive motors in desired velocities. All the raw sensors value from sonar, gyroscope, compass, wheel encoder were processed and delivered to computer on its request. Figure 18 shows the actual Arduino mega board.



Arduino Mega Microcontroller (Gautam 2014)

Inter-Integrated Bus

I2C stands for Inter-Integrated circuit. It is quite popular in embedded system because of its simplicity, built in addressing and multi drop functionality. Pair of wires; one containing data and other containing clock is used to establish two way communications between multiple devices. Data transmission rate up to 400Kbps is supported by modern I2C device. Fig.19. shows devices that were connected using I2C bus in this project.

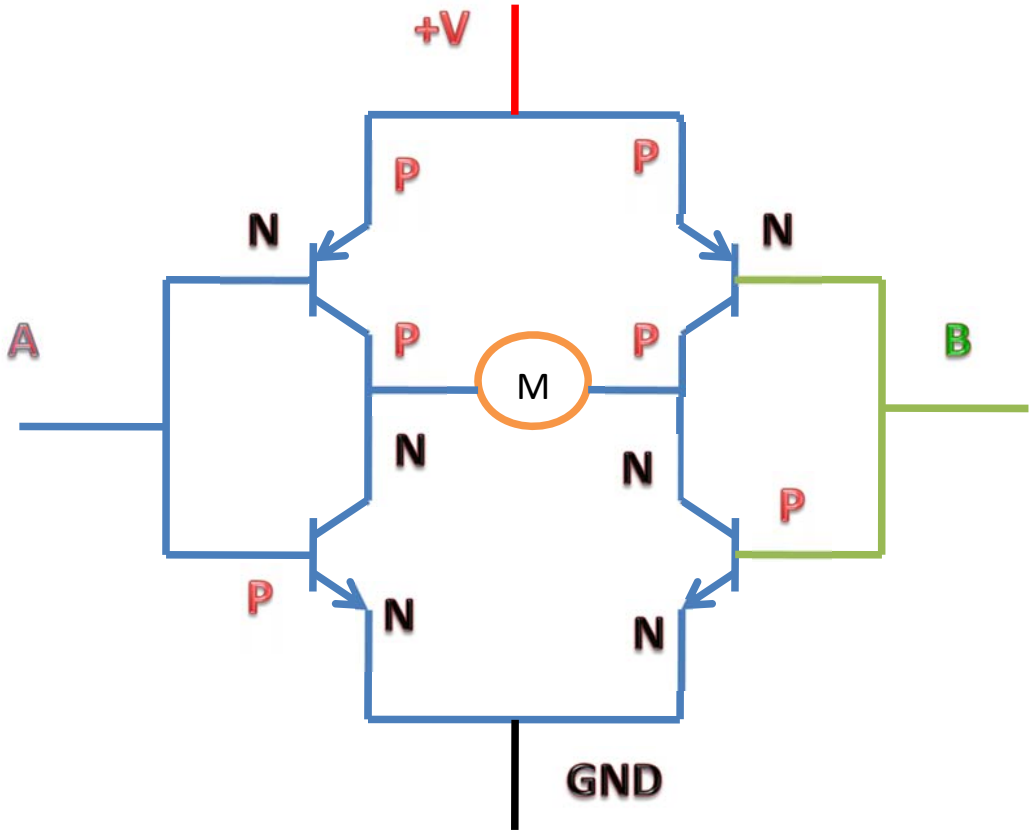


I2C bus device (Gautam 2014)

As shown in figure20 data was transmitted in a packet and each packet contains: start condition, slave address, slave acknowledgement, actual data, master acknowledgement and stop condition.

Motor Drive Circuit

Motors used in this project consume relatively high current that is 2 ampere per motor. The output of microcontroller cannot supply such a large amount of current. Therefore, readymade motor H-bridge made of a pair of metal oxide field effect transistor was used. Primary function of H-bridge is to switch high power device into low power or vice-versa. Besides switching application, it is also used to change direction of motors and circuit isolation.



H-Bridge Schematic (Gautam 2014)

Figure 21 shows the basic schematic of MOS FET H-bridge.

DC Motor

Power window motor of a car was used to drive wheels of the robotbecause it has relatively large torque in a compact size. It is a type of permanent magnet direct current motor (later PMDC motor) with a rotor shaft connected to worm gear. By the definition PMDC motor consists of radially magnetized permanent magnets, which are attached on the wall of motor housing forming a stator core to produce field flux and a rotor having dc armature with a commutator segments and brushes (Electric Motor, 2014). Figure 22 (a) shows the actual motor used in this robot and the torque relation.

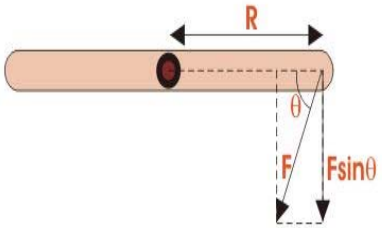


Figure (a): Power Window Motor Figure (b): Torque relation

PMDC motor (Electric Motor 2014)

According to figure 22 (b), the equation of torque is given in equation (4):

$$T = FRSin\theta \qquad (4)$$

Where F is force in linear direction, R is a radius of rotating object and θ is the angle made by force F with radius R vector.

3.2 Software and strategy

This chapter aims to explain about vision based servoing. The term means that the control of the robot is based on the data acquired from camera. Sensors data discuss so far was easy to process but provided very less amount of data. It was difficult to estimate the region or the environment being sensed on. Limited information about environment results in poor navigation through obstacles. Therefore, a sensor which can provide large amount of data about the sensed environment was necessary. Complementary metal oxides semiconductor (later CMOS) and charged coupled device (later CCD) technology based vision sensors were so far best at sensing the environment. For this project, the authors had used Logitech HD webcam C615 was a CCD based fluid crystal camera (Logitech 2014). Pair of vision sensors was used to acquire image from crops row. Figure 33 shows the acquired images and images after processing when using single camera at the centre of the chassis.

Row Detection (Gautam 2014)

For effective navigation the robot must be able to extract meaningful information from environment at any situation, it must be able to keep track of its position relative to world coordinate, it must be able to determine optimal path to reach the goal and finally it must have control over its actuators. To accelerate navigation and minimize errors vision sensor was placed in such a way that the field of view was limited to one row for one sensor. The data acquired from this sensor was used for path planning and navigating the robot within crops field to reach the set point.

Machine Vision

This section is aim to explain the basic of image acquisition, image processing and image features extraction. Dependency of image quality on different factor is explained with an aid of figure. Digital image contains very large amount of information, handling such a large amount of data need a very large computational power. However, all the information contained in a digital image is not necessary. A very small portion of information is enough to extract meaning full information from it. The practical aspect of extracting image features is explained in different subs-headings.

Image Acquisition

An image acquisition is a process of capturing scene and representing it in a user defined format. Nowadays, images are stored digitally. It is possible to capture an image digitally if an object reflects light energy received from sun or any light emitter (Moeslund 2012). As shown in figure 23, a suitable optical system was used to capture scattered rays and an array of photoreceptors (photo diodes) were used to measure light intensity of reflected light energy from that object.

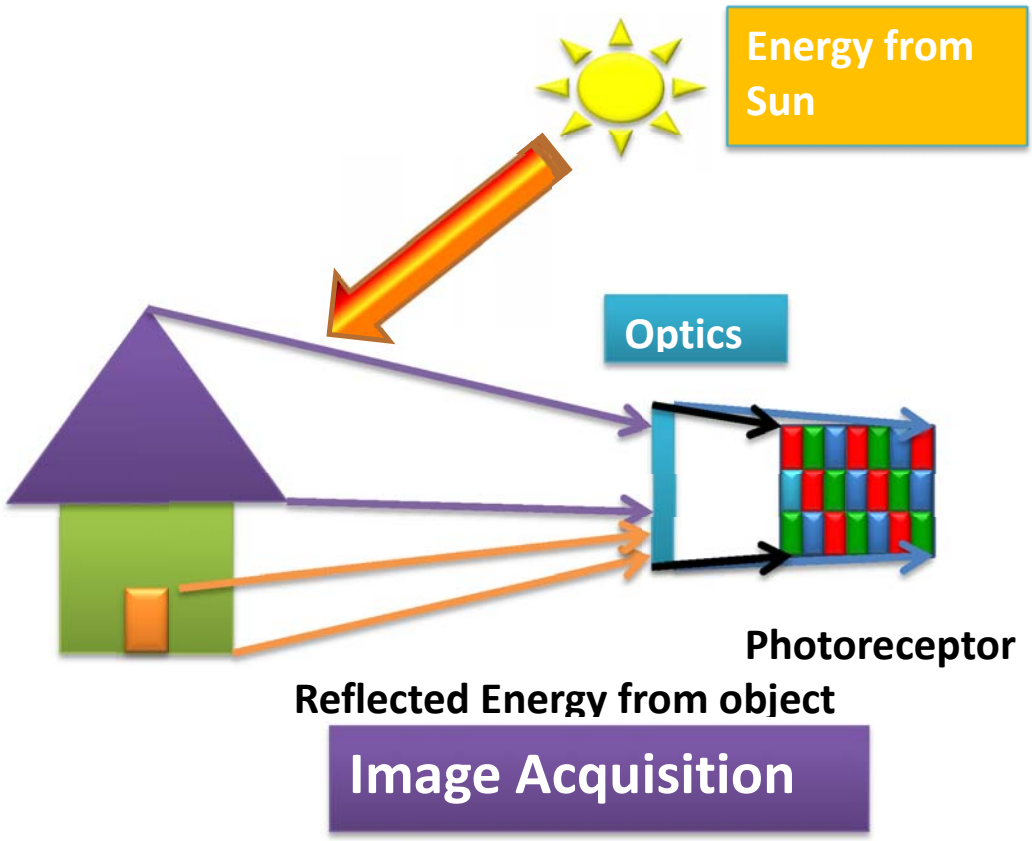
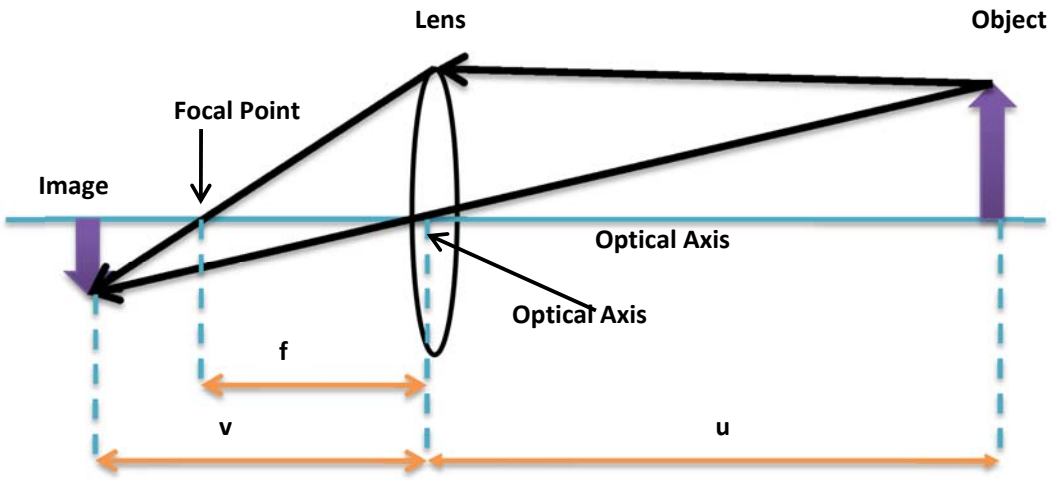


Image Acquisition (Gautam 2014)

Smallest element of digital image is called as a pixel. Each pixel stores the intensity of electromagnetic waves. If an image is a gray scale type then its intensity value is represented from 0-255 where 0 represents black region and 255 represents white region (Moeslund 2012). And in case of colour image, original image is filtered using red, green and blue filters. Each receptors measure the intensity of light which has passed through specific colour filter. Combination of these three basic colours is used to represent all other colours. Digital images largely depend on optics type, pixel value and reflected light from objects. In an outdoor environment the intensity of light changes unexpectedly. So an extra precaution must be taken to filter this noise.

Optics

When light falls on any object, some of the lights are absorbed by the object and some are reflected. The reflected rays gets scattered most of the time that drops off the quality of image so, to get better quality image a suitable lens must be used. It is because lens can focus multiple rays of light coming from same point on a single point (Moeslund 2012).

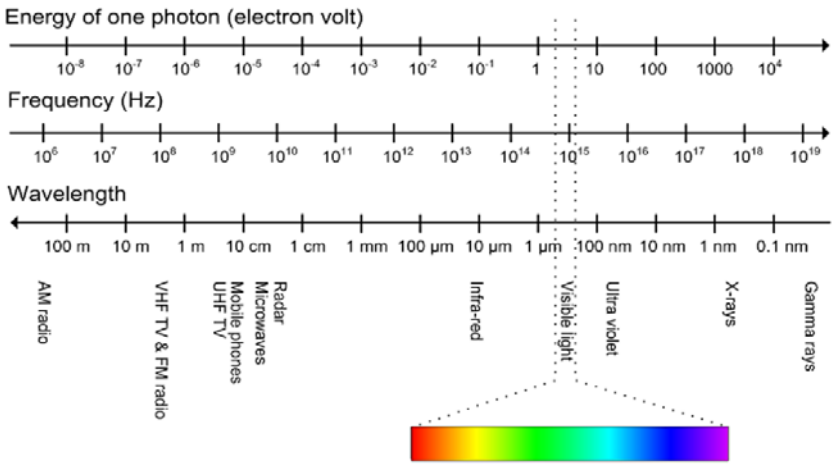


Fundamentals of Optics (Gautam 2014)

Focal length and the aperture are the basic parameters of lens. As shown in figure24, focal length play key role in magnification of image and the intensity of light is regulated through aperture. The main disadvantage of using optical system is that it deforms images. However, it can be corrected using suitable algorithm.

Illumination

Illumination is an important factor in machine vision because the quality of an image depends on the intensity of light energy received by photo receptors. If the intensity of reflected light is less than the minimum threshold then the image will be of poor quality where as if the intensity of light is above the maximum threshold then the image will be very bright (Moeslund, 2012). Hence, suitable lightning must be used for better image resolution. But the light source should not be behind the image otherwise it will capture dark image.



Electromagnetic Spectrum (Moeslund 2012)

Figure 25 shows the light spectrum. In an outdoor environment the intensity of light changes unexpectedly. To improve image resolution in poor lightning condition, suitable lightning must be added. As we know that defuse lightning system can illuminate an object from every directions so, that the photo receptors can capture very tiny details. Hence, it is a good idea to use defuses lightning system. When the intensity is very high then a suitable filter must be used to regulate the threshold and to prevent an image from being glared. In our application we had used polarised glass to improve image quality.

Digital Image Processing

Digital image is a two dimensional representation of scene and it has certain intensity value. Mathematically, it is represented as function of f(x, y) where, x and y represent position of pixel in 2D plane (Young, Gerbrands and Vliet, 2007). Image acquired from any sensor are subjected to disturbances. It is important to filter those disturbances as it affects computations which are based on these data. There are several known methods to minimize those disturbances. The process of minimizing disturbances and extracting useful information from raw data is called image processing.

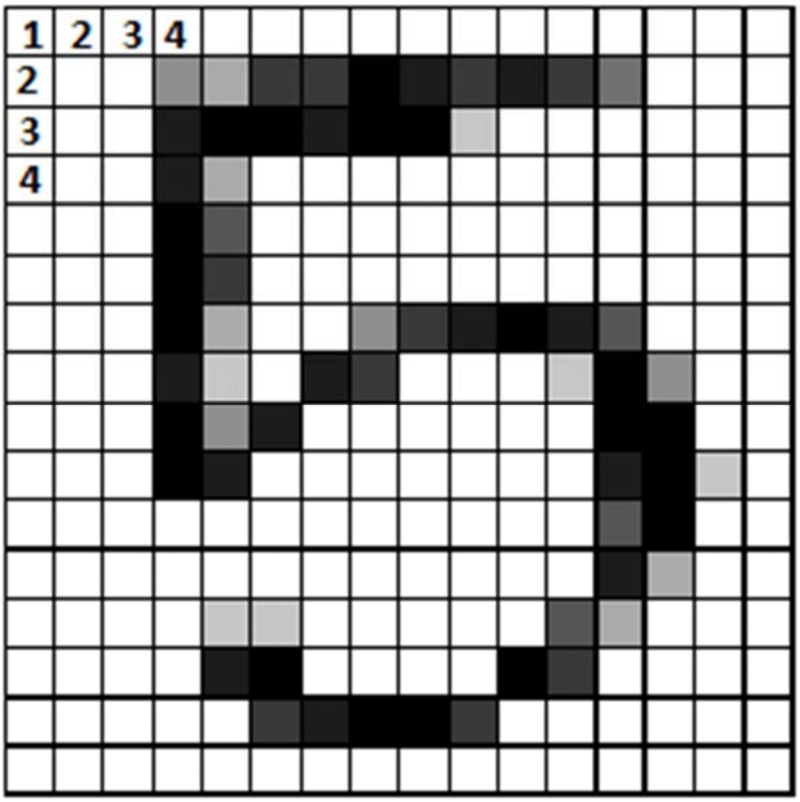


Image representation in digital format (Young, Gerbrands and Vliet 2007)

From the figure 26 we can determine the coordinate of each pixel. For example first pixel with intermediate gray scale value has a coordinate (x=2, y=4). We can also see that the image is not vivid; in order to enhance this image we can use different image processing methods such as sharpening, de-blurring, adjusting contrast, brightness and highlighting edges.

Image Features Extraction

A digital image consists of large amount of data. Extracting all information from such source needs large computation power and time. In this project the robot had to navigate through maize plant rows and find weeds which were represented by yellow colour golf ball. To minimize computational power and time only certain information was extracted such as colour, texture and shape. This information was sufficient to navigate the robot through maize row and determine relative positions of yellow coloured golf balls.

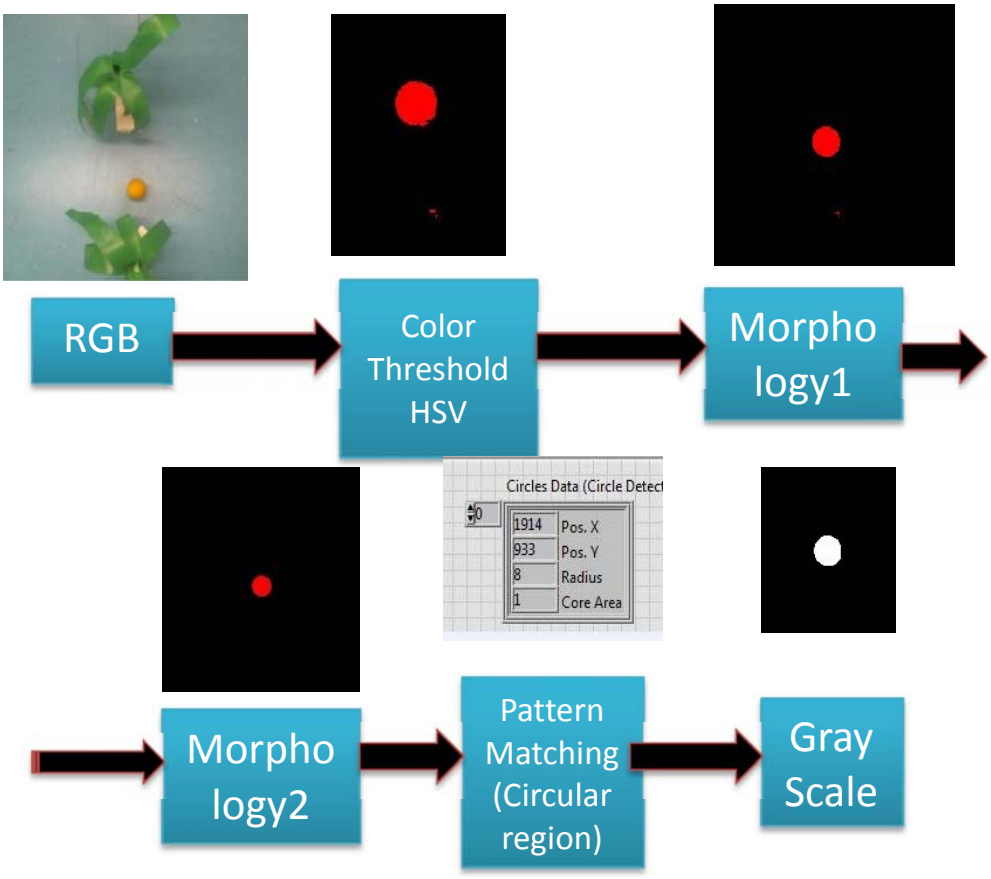


Image feature extraction process (Gautam 2014)

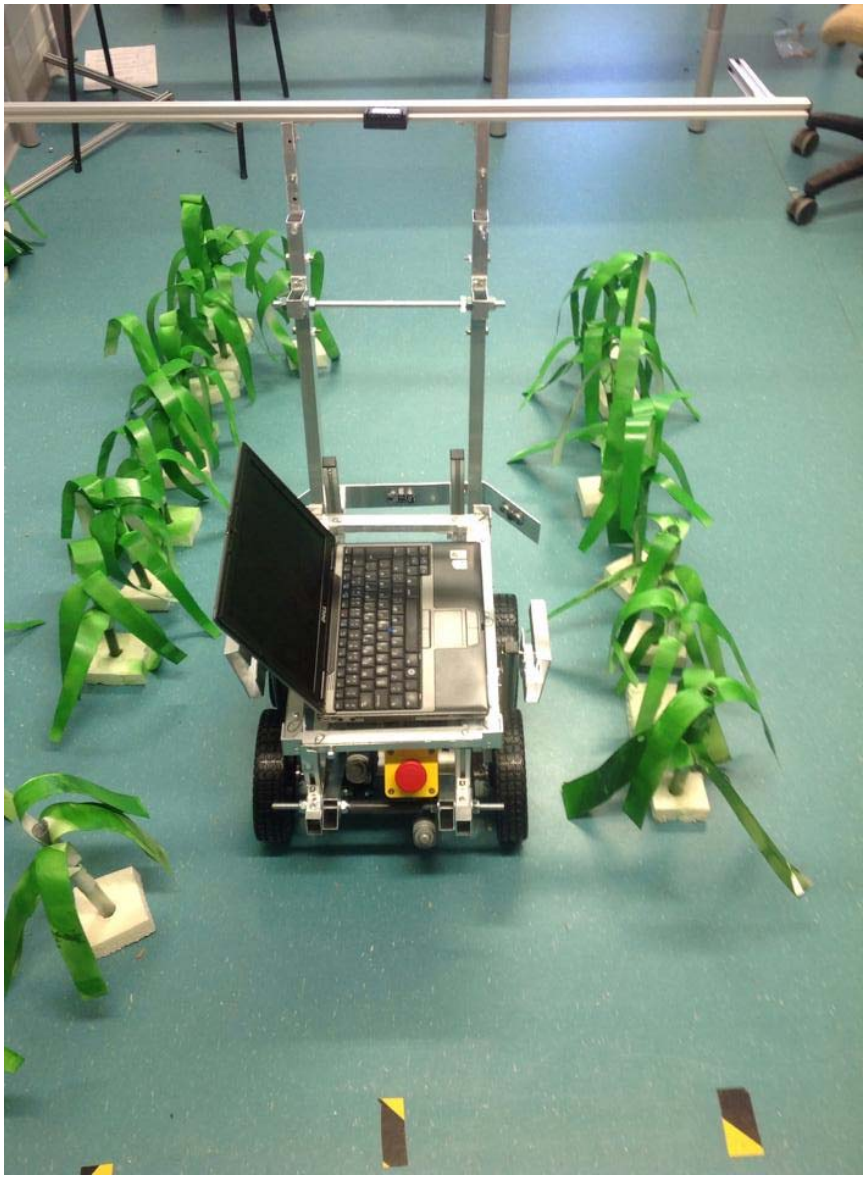
Figure: 27 shows image feature extracting process using NI LabView vision tool. In this example, yellow colour golf balls were used as a sample objects. The main task was to identify these balls using a camera in an agricultural field. Firstly, RGB image was

captured with camera attached on the robot. HSV reference was used to threshold the image (Smith and Chang 1995). Resulting images contain a lot of holes and irregular surfaces. In order to improve texture of image, morphology tool was used to fill those small pores on the images. Again Morphology tool was used to remove small regions which are below rated threshold so that unwanted reason can be removed. To identify the shape, a circular region was searched throughout the image with predefine range of diameter and also the coordinate of circle was determined. Finally, this image was converted into gray scale image to display on GUI. Program block of ball detection is attached in appendix 4.

Reactive Planning and Navigation

This section deals with robot's optimal path planning and collision free navigation to the goal. The robot develops the map of surrounding autonomously from the information gathered through sensors. This map is used for effective planning of path to reach the goal. Different methods are used for path planning and navigation in robotics. Such as Road map, Cell decomposition and potential field (Corke 2011, 89-91). Among which road map based, map building was used in the project.

Road map was used because it was simple and easy for computing. The robot had a freedom to move on free space which leads to the shortest displacement to the goal. Sonar sensors on left hand sides and right hand sides of the robot provide the distance information to the row follow controller. This information was used to keep the robot always in middle of crops rows. Any deviation on the robot position was corrected with its reference. In addition to this, a camera was used to assist the navigation. Position of the camera was just above the maize row (as shown in figure 34). This position was selected because the crops rows were curved in nature. Change in the orientation of the robot was synchronised with instantaneous change in crops row. A closed loop PI controller was used to keep track of change and to produce a control output to meet the changes.



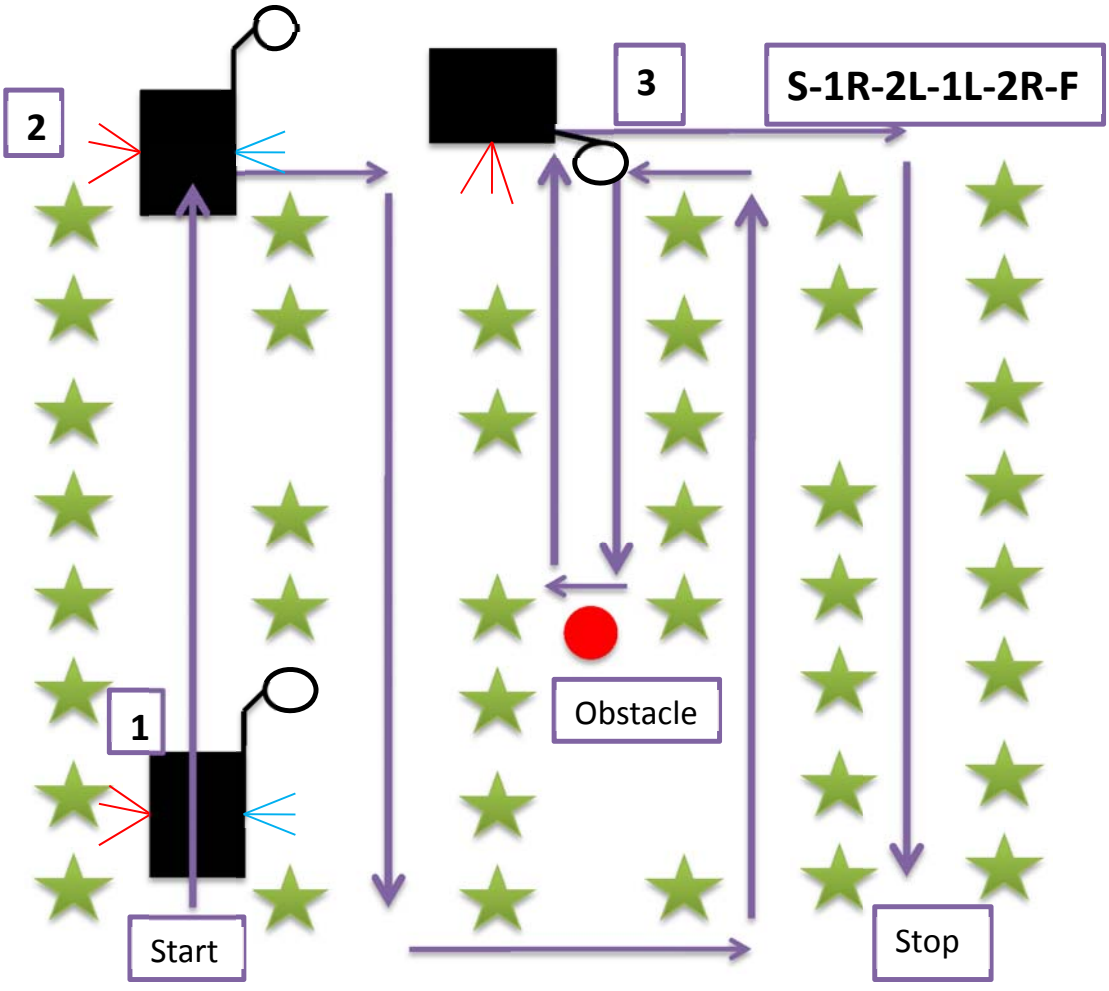
Navigation based on crops row (Gautam 2014)

Reactive planning was used in task 1 and task 2 in FRE 2014. In task 1 input from camera was only used for navigation because in this task all rows had equally distributed maize plants and there was no missing plant. However, in case of task 3: several maize plants were missing in maize row and the maximum gap between two plant can be up to one meter in either of the side. Hence sonar sensor reading was also used to determine the presence of maize plant in opposite row. Program block of basic navigation is attached in appendix 5.

Map base Planning and Navigation

Map based planning and navigation is the best way to reach from point A to point B in any environment (Corke 2011, 91-105). However a predefined map cannot provide every bit of information about the dynamic environment. To overcome this issue artificial intelligence was developed. Whenever a predefine map and a real environment contradicted than in this situation light the robot was able to make the appropriate decision to overcome reactive navigation problem.

This module was especially designed for task 2. In this task, a predefined code of turns was given to each team. For example: Start-3Left-0-2Left-2Right- 1Right-5Left-Finish (FRE 2014). The robot had to navigate through maize row following that pattern. Traffic cones were placed as an obstacle in the middle of the robot path. The presences of such obstacles were not defined on the map. Hence, the robot can encounter such obstacles at any point within the field. To overcome this challenge, the following approach was carried out.



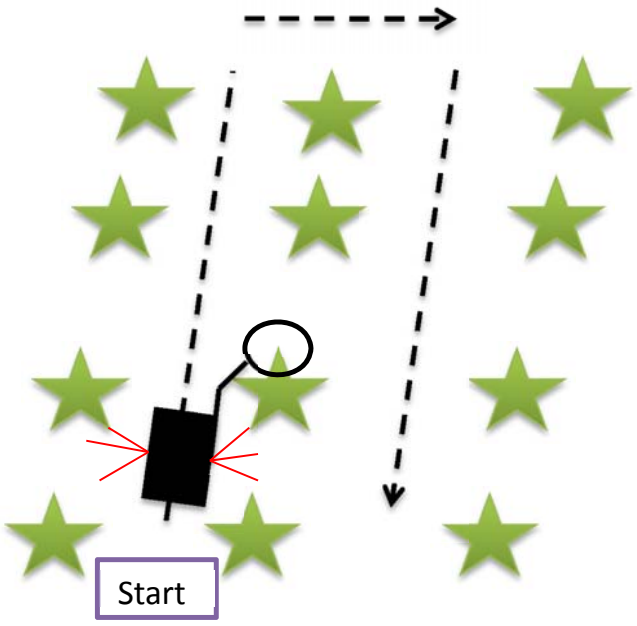
Map based navigation for task 2 (Gautam 2014)

A code of turn (map) was designed and stored in the robot's memory. It consists of a start position; row length value, clockwise and counter clockwise turns after an 'x' number of rows (where x represents a natural number). The robot was designed to navigate using the data from camera. But a maize plant was missing in some parts of the rows. Because of that the sonar sensor reading and a distance measurement from wheel encoder were used for navigation in task 2. If the maize plant was not detected within an effective field of view of the camera (as shown in state 1 in figure 35) then the data from opposite sensor was checked. If the data from the opposite sonar senor was true then the follow row control module was continued otherwise the "Change Direction" control module was enabled. Before turning the robot was programmed to move 30 cm forward and the reverse turn 30 cm if maize was not detected on that move. This action was carried out to detect the end of row. For better approximation wheel encoder value was checked before switching to change the direction mode. Each row was counted using inner sonar sensor and if the row was not detected within 0.75m then one row was counted after every 0.75m of distance value from wheel encoder.

Finally, map was updated if any obstacle was detected within crops field. New path was determined after reverse driving until the wheel encoder value reaches back to zero or until the robot reaches to the end of the row. Updated path was followed to navigate through maize plants and to reach the goal.

Follow Rows

First and foremost control module developed was follow row. This module was used to navigate the robot in between two maize plant rows. For this task data from camera and sonar sensor was used.



Follow Row (Gautam 2014)

As shown in figure 30 camera was placed just above the crops field so that the robot can follow crops row to reach the goal. Sonar sensor attached on the side was used to keep the robot equidistance from the rows and the heading angle was obtained from compass attached to it. This module was used in all the task of FRE with other modules. Wheel encoder signal was used to store the total distance travelled before the change of controller.

Change Direction

In this module, turning mode was defined. This mode was used to take clock or counter clockwise turn as soon as the robot reaches end of row in conjunction with distance value from wheel encoder. This mode was used in all tasks when the maize plant was not detected within the field of view of camera. Data from magnetic compass was used to align the orientation of the robot with respect to earth's magnetic north. Angular acceleration was closely monitored for smooth turning. Turning radius was calculated from the formula shown in equation (5) and described on "differential speed steering control for four wheel independent drive electric vehicle" (Wu, et al. 2013).

$$Ro = \sqrt{ICRx^2 + ICRy^2} \qquad (5)$$

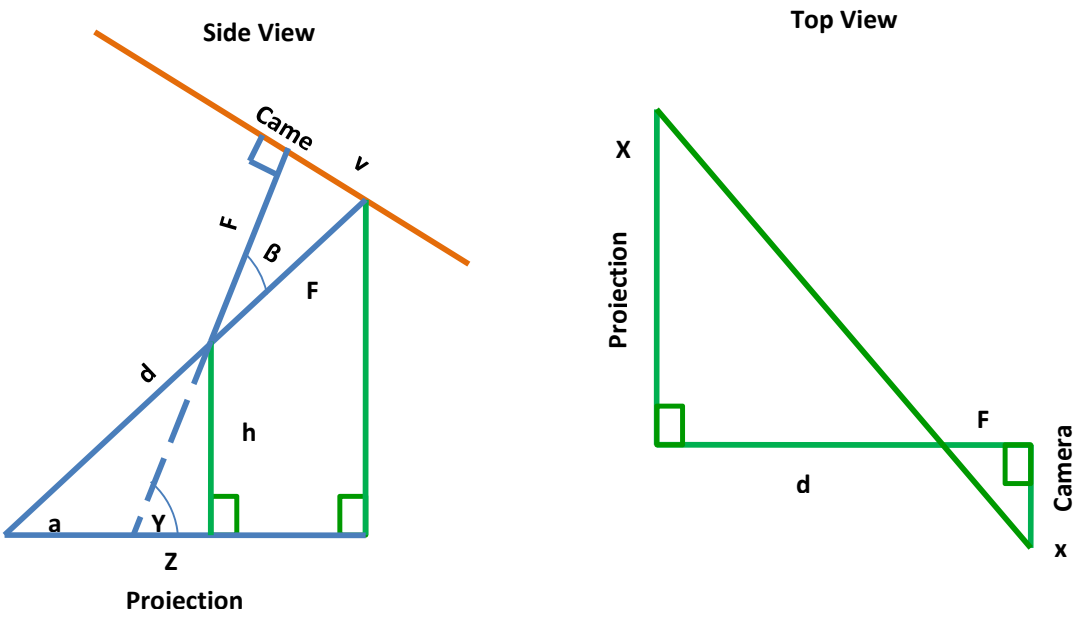
Where,

Ro is radius of turn

ICR is an instantaneous centre of rotation.

Locate Goal

The main aim of this control module was to locate the absolute position of weed plants randomly placed in maize field. This was achieved by using RTK GNSS system to know the absolute position of the robot on the crops field and relative positions of weed plant were determined by processing images captured from camera.



Resultant vector to determine weed plants from known height

Whenever a weed plants were detected within the effective field of view then its centre was calculated using NI vision tool and a buzzer was activated as a successful identification. The coordinate transformation algorithm was used in LabView. It was assumed that the height of the camera and the weed plant were known and were kept constant throughout the test. Thus it was possible to project the pixel describing the centre of mass of the object down to a plane that was parallel to the ground. It was also possible to obtain desire height by using pinhole optics as an approximation.

Figure 32 shows the resultant vector that was calculated between robot's camera lens centres to weed's centre point which was based on following equations:

$$x = \left(\frac{x_{px}}{w_{px}} - \frac{1}{2}\right) w_{mm} \quad (6)$$

$$y = \left(\frac{1}{2} - \frac{y_{px}}{h_{px}}\right) h_{mm} \quad (7)$$

$$Z = \frac{h}{\tan \alpha} \quad (8)$$

$$\beta = -\tan^{-1} \frac{y}{F} \quad (9)$$

$$d = \frac{h}{\sin \alpha} \quad (10)$$

$$F' = \sqrt{y^2 + F^2} \quad (11)$$

$$X = \frac{xd}{F'} \quad (12)$$

Where Wwmm and hmm are the dimensions of the camera sensors,

Wpx and hpx are the dimensions of the image pixel,

γ is the tilt angle of the camera

'h' is the difference in height between the weed plant and camera

And F is the focus length. (Hafren, J., Alaiso, S., Karppanen, E., Kostiainen, J., Rannisto, J., Sosa, R., Valli, A. And Virta, A. 2012, 35-37.)

4. Conclusion

The authorss have closely observed traditional method of agriculture in his home town. And the closeness towards traditional method of farming has dragged his interest for improving it. Thus, working for FRE 2014 provided a great opportunity to design and control an autonomous robot. Different research paper on implementation of autonomous robot for precision farming was examined to improve the stability, manoeuvrability and robustness of robot.

Test results obtained from wheel encoders, sonar sensors, IMU, were carefully studied. Changes were made based on these data. Initially, the authors had implemented IR based distance measurement, but the test results were found to be highly affected by sunlight. Thus, sonar sensors based distance measurement was used. However, the information acquired from sonar sensors was not enough to perform the entire task. Because, the amplitude of reflected echo from maize plant was not strong enough if the plant had small stem and few leaves. So, to collect sufficient data from the dynamic environment, a camera was used. Camera was chosen as a best option because; the programming task could be simplified to one colour extraction. As the green coloured maize plants row could be easily identified from brown coloured soil by using colour separation technique. The robot was programmed to follow the green colour row for safe navigation. The performance of the robot after using camera was found to be very good.

Digital image obtained from camera was used for navigation. It consists of large amount of data. Microcontroller available in market was not capable of computing such a large amount of data to produce desired output. Hence, laptop was used for image processing. And Arduino Mega was used as the interface board to connect sensors and actuators. Serial communication was established between laptop and microcontroller. Use of laptop had added flexibility to connect and control this project remotely. Beside the weight of laptop, the combination of it and microcontroller was found to be perfect.

Before implementing any module, it was well studied and tested in the controlled environment. A test field was constructed to mimic a real maize field. Maize plant was erected with a plastic tube as a stem and strip of green paper as a leaves. Different control modules designed for different tasks were tested in it. After a series to test it was found that none of the navigation algorithms was effective enough to complete all the levels of FRE 2014. Because of which, those navigation algorithms were switched based on the scenario to achieve effective navigation. This method had helped considerably to reduce the number of errors. As a result, chassis of the robot was able to absorb vibration created by uneven terrain; oscillation of the robot was negligible, it was able to make clock wise as well as counter clock wise turn to enter a new row in a single move, it was able to avoid obstacles, and finally; it was able to catch the set points with a very small delay to navigate through crops row.

This project was only tested in an artificial environment. There might occur a change in the performance while testing it in outdoor environment. There is always a void between prototype model and a real system. The authors had boosted his best to fill this void. Despite of satisfactory working of robot there are several possibilities to improve it. The authors had faced difficulties in calibration of low resolution sensors. For example sonar sensor reading was highly affected by noise. Thus, the authors suggests using high resolution sensor and to replace sonar sensor by laser range scanner. Laser range scanner has very good resolution and it is robust. Furthermore, the authors suggests implementing stereo vision and simultaneous localization and mapping (later SLAM). Stereo vision and SLAM helps to build a 3D map of the environment and which can be updated real-time. This helps to develop artificial intelligence in robot.

5. References

Field Robot Event 2014- Robot Fun and Creativity. University of Hohenheim, Instrumentation and Test Engineering. Accessed 12th March 2014.

https://fre2014.uni-hohenheim.de/tasks

Wu, X., Xu, M. and Wang, L. 2013. International Journal of Materials. Mechanics and Manufacturing Journal, pdf-file, vol.1, no.4. Accessed 14th March 2014.

Link: http://www.ijmmm.org/papers/077-A009.pdf

Foesseol, A. Radar Sensor Model for Three Dimensional Map Building. Proc. SPIE, Mobile Robotis XV and Telemanipulator and Telepresence Technologies VII. Accessed 14th March 2014.

http://www.ri.cmu.edu/publication_view.html?pub_id=3399

Oslon, E.2009. A Primer on Odometry and Motor Control. Massachusetts Institute of Technology. Electrical Engineering and Computer Science. Accessed 19th March 2014.

http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-186-mobile-autonomous-systems-laboratory-january-iap-2005/study-materials/odomtutorial.pdf

Caruso, M. 1997. Application of Magnetoresistive Sensor in Navigation Systems. Minnesota State: Honeywell Inc. Accessed 24th March 2014

http://inertial-solutions.us/pdf_files/sae.pdf

Hazry, D., Sofian, M. and Azfar, A. 2009. Study of Inertial Measurement Unit Sensor. University Malayasia Perlis. Accessed 28th March 2014

http://www.researchgate.net/publication/228995014_Study_of_inertial_measurement_unit_sensor

Vincent, D. 2013. Accurate Position Tracking using Inertial Measurement Units. Miami University. PNI Sensors Corporation. Accessed 3/5/2014

file:///C:/Users/Samrat/Downloads/Accurate%20PositionTracking%20Using%20IMUs%20(1).pdf

An Introduction to GNSS 2014. NovAtel Inc. Accessed 28th March 2014

http://www.novatel.com/an-introduction-to-gnss/chapter-1-gnss-overview/

Land, B. 2012. Cornell University. School of Electrical and Computer Engineering. Video lecture 28—TWI(I2C)

https://www.youtube.com/watch?v=7yRoR7w82SA

(Motor H-bridge, 2011) Pololu Site Editor: Albertolg. Accessed 7th April 2014

http://abottravel.blogspot.fi/2011/11/h-bridge-qik-2s12v10-dual-serial-de.html

(Electrical Motor 2014a). Electrical4u. Site Editor:

Accessed 4th April 2014. http://www.electrical4u.com/

Moeslund, B. 2012. Introduction to Video and Image Processing. Underdradute Topics in Computer Science. London: Springer-Verlag. Accessed 6th April. DOI 10.1007/978-1-4471-2503-7_2

Young, I.,Gerbrands, A. and Vliet, L.2007. Fundamentasl of Image Processing. Netherlands: Delft University of Technology. Accessed 6th April

ftp://qiftp.tudelft.nl/DIPimage/docs/FIP.pdf

Siegwart, R. And Nourbakhsh, I. 2004. Introduction to Autonomous Mobile Robot. London: MIT Press

Hafren, J., Alaiso, S., Karppanen, E., Kostiainen, J., Rannisto, J., Sosa, R., Valli, A. And Virta, A. 2012. RoseRunner Field robot Event 2012. Aalto university and University of Helsinki. Department of Automation and Systems Technology, Department of Engineering Design and Production. And Department of Agriculture Sciences.

Phillips, C.L. and Harbor, R.D.(2000), Morden Control Engineering, Prentice Hall, Upper Saddle River, New Jersey.

Logitech 2014, Technical data sheet. Accessed 20/4/2014

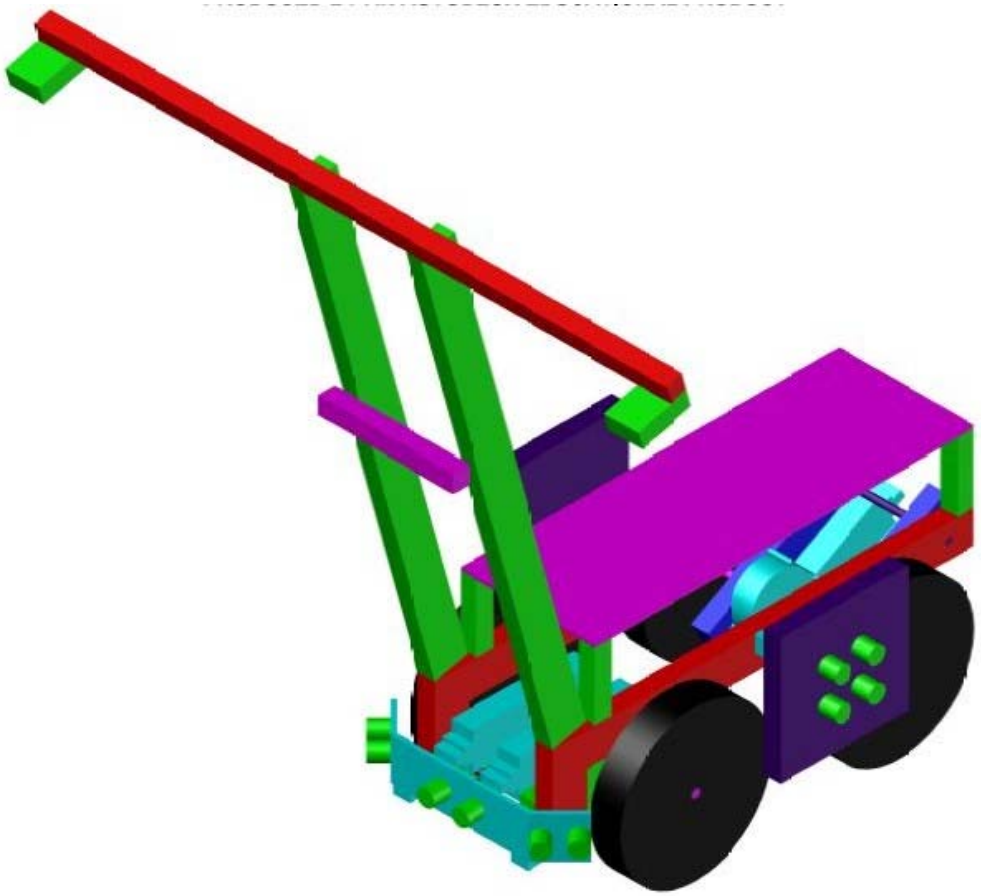
http://images10.newegg.com/Manufacturer-Brochure/Manufacturer_Brochure_26-104-469.pdf

Corke, P. 2011. Robotics Vision and Control. Chennai: Armin Stasch and Scientific Publishing Services Pvt. Ltd.

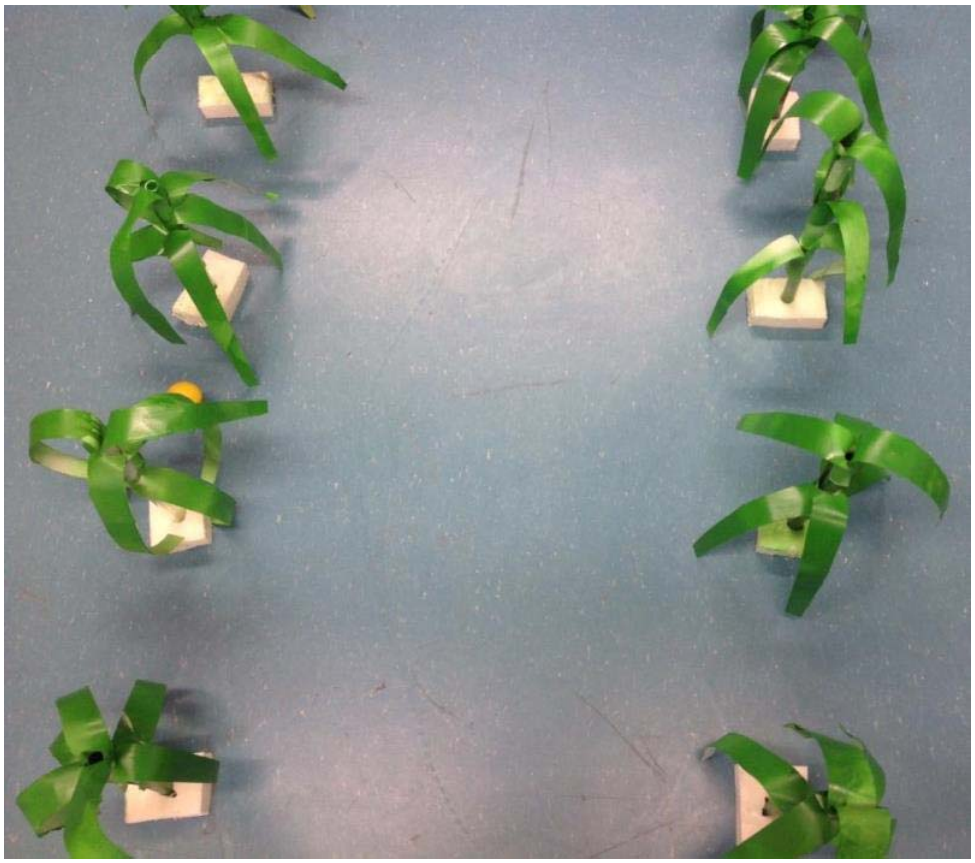
Smith, J. and Chang, S. 1995. Single Colour Extraction and Image Query Columbia University, Center for Telecommunication Research, Image and Advanced Television Laboratory. Accessed 9th April 2014

http://www.ee.columbia.edu/ln/dvmm/publications/95/smith95b.pdf

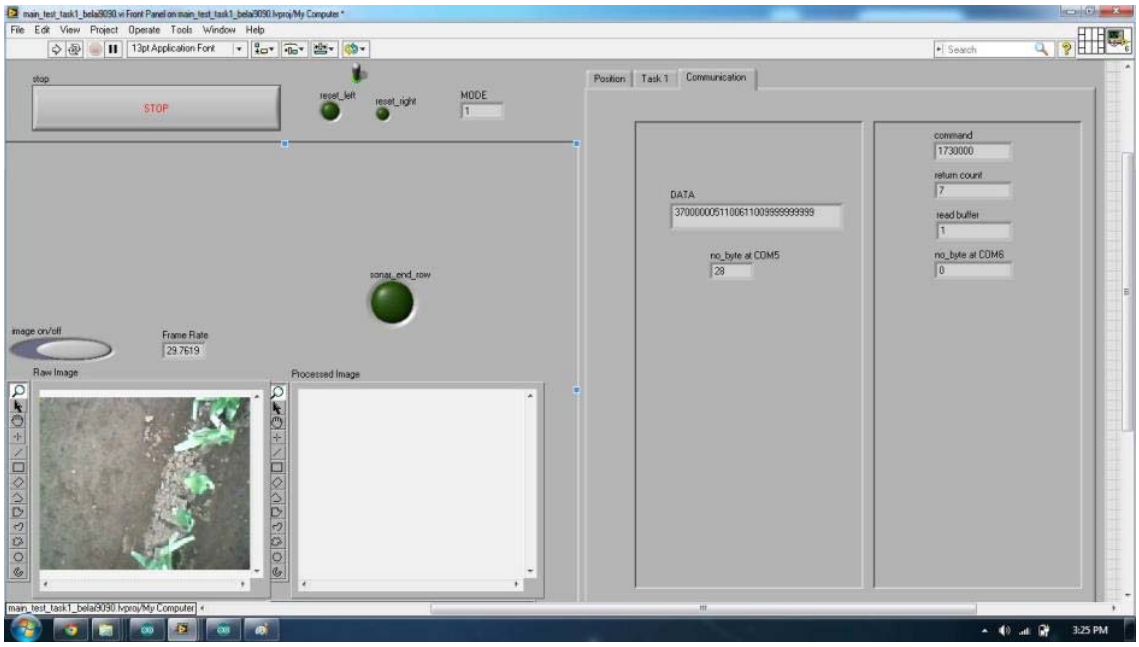
Appendix 1: Modelling of Agro-Bot

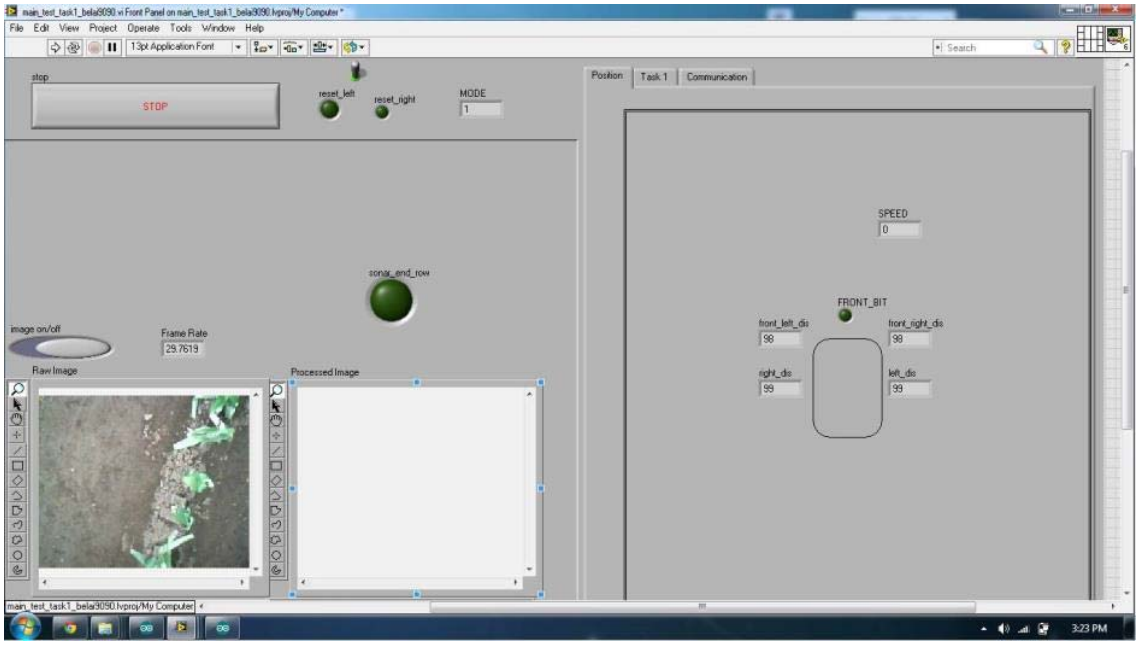


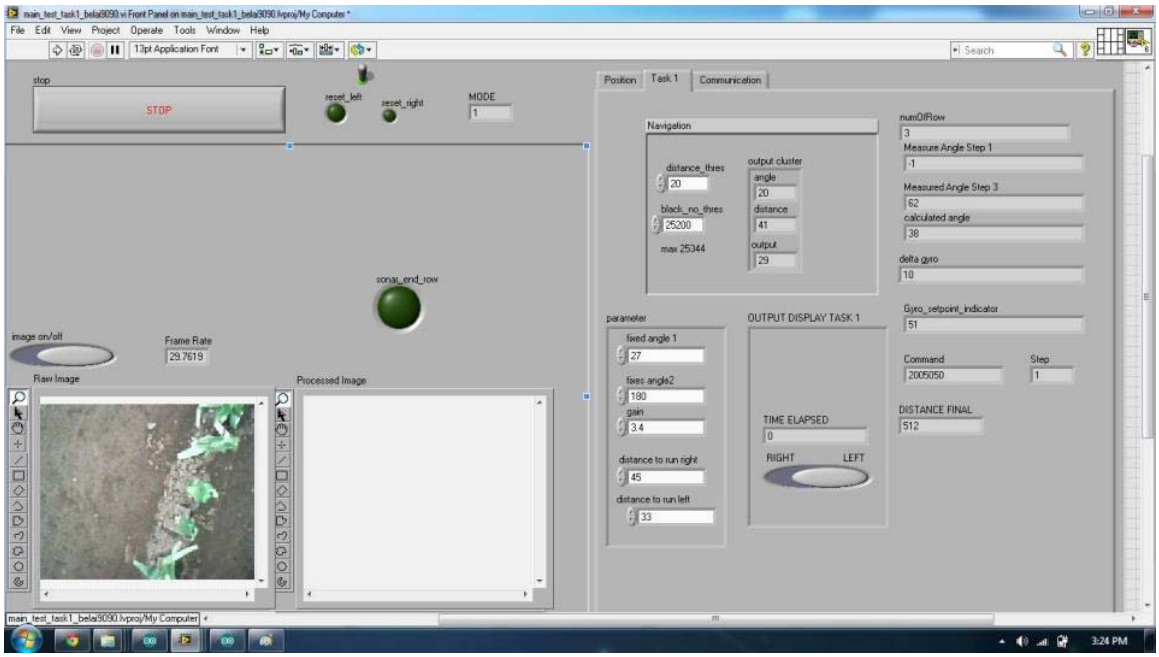
Appendix 2: Indoor Game Field



Appendix 3: Main Program Print Screen







Appendix 4 Follower for irrigation

“Banat” an autonomous agricultural robot

Cristian-Leonard Fitz

Universitatea Politehnica Timisoara (UPT), Mechatronics Department, Timişoara, Romania

1. Introduction

In a period in which everything must be better and better and the price as low as possible, people came to the conclusion that robots can successfully meet these demands. That is why, their use is increasing in the last period of time.

The agriculture cannot meet the present needs without catching up with the modern technology, thus the emergence of automatic control and production systems and of the agricultural robots is already a reality.

This paper deals with the design and production of the first autonomous agricultural robot in Romania realized by the “Banat” team, now at the third review.

The achievement was completed after more than 1,600 hours of work, under the supervision of the Associate Professor Sorin-Tiberiu Bungescu and according to the valuable suggestions of Professor Dumitru Tucu and Lecturer Sorin Nanu and the help of my colleagues Catalin-Ciprian Adrian Pisoiu, Eniko Toth, Paul Negrila, Caius Simandan and Catalin Almasan.

2. Mechanics

The working speed of the robot in the interval 5-8 km/h.

The dimensions of the robot to be able to navigate in the cornfields hoes with the distance between the rows of 70-75 cm and at the same time to avoid damaging the plants, were set at:

-Length = 400 mm;

-Width = 400 mm;

-Height = 200 mm.

The maximum mass of the robot was estimated to be no more than 12 kg, and additional equipment included attached to it.

After considering the requirements to be met in carrying out the robot and to the readily available and outlined above, have made the following choices with regard to the solution of each system.

The robot

Type of frame for the achievement of this robot, is the scale, made of aluminum profiles to ensure the necessary characteristics in terms of rigidity and strength.

MK Profiles is the company that gave us very good quality products, which have covered all our needs both in terms of the profiles and connecting them, providing for joint bottom screws, washers and nuts for assembling and quality meeting the needs of our rigidity.

The running System

As a constructive alternative to the running system has opted for wheels with tires for the Elimination of the additional forces that may arise in the case of the tracks would be extra load to the engines.

Choosing the most appropriate I thought to be S GT2 wheels Rims CHROME COMPOUND on WARLOCK used in car models 1: 8 scale used in climbing and competitions that are recommended on any type of surface.

Fig. 2.8 Wheels with tyres

Steering System

Steering system with different angular speeds of the wheels was chosen to reduce the complexity of the robot, the dangers of failure as well as the price of carrying it.

It shall consist of a pinion of a bicycle for each wheel, the transmission of the movement being performed by a bicycle chain on each side. Noted that the number of teeth is equal for each wheel, so that the angular speeds of the two wheels (both head and run) to be equal.

Fig. 2.3 The steering mechanism

Transmission

Due to the space available and the necessary characteristics of the transmission from the engine to the wheels, the chosen transmission is a reducer with gears.

This swing is made of 4 gears the two transmission reports to decrease the speed of the wheel up at 225 RPM., increasing the torque up to 23 times.

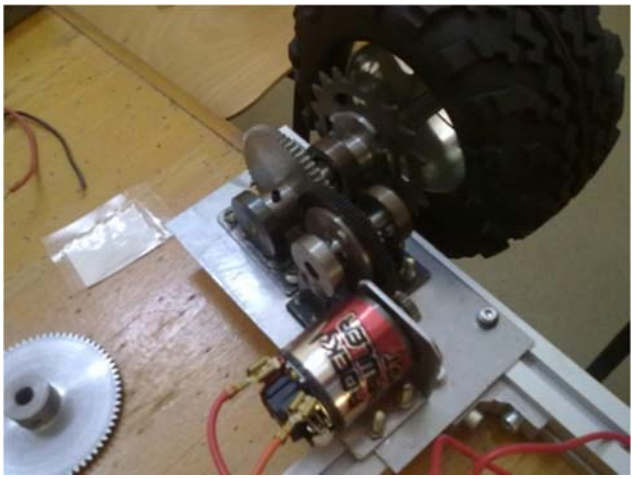


Fig. 2.4 Transmission

Propulsion

Propulsion of the robot will be carried out with the help of two electric motors of direct current. The power that we need to develop the two engines was calculated, resulting a required power of $0{,}089kW$.

Engine model chosen is Absima 540 x 80T x 2 PCs.

Efficiency of the transmission is:

$$\eta_T = 0{,}8$$

Maximum power transmitted to the driving wheels is:

$$P_{er} = P_{Mr} \cdot \eta_T [kW]$$

$$P_{er} = 0{,}16 \cdot 0{,}98 = 0{,}157\ kW$$

In this way the power requirement is satisfied with the two engines.

Technical specifications of engines are listed below:

Technical data:

Supply voltage 7,2-7,4 V

Spire	80
Speed	5300 s^{-1}
Power	80 W
The shaft length	11 mm
Shaft diameter	3,175 mm
Dimensions	(Ø x H) 35 mm x 35 mm
Weight	165 g (ca.)

Fig. 1.8 Engine

3. Conclusion

This robot has been a challenge, completed after a race against time, along a over 1600 hours of work. The help came from colleagues at UPT, both those in mechanical workshops, as well as those from automation, was highlighted by the fact that the robot has stuck and has had great success at international competitions in which he participated.

Due to the fact that the robot has fulfilled all the tasks for which it was designed is bound to delight me and motivate us to continue research and development in this area.

After those seen during the design, and testing and participation in competitions, it is obvious that this branch is growing from year to year, the future of agriculture will be writing soon by robots.

From the point of view of the robot in this work, it is only at the beginning, being the only one of its kind in the territory of our country and of course it needs improvement. What improvements will be made are so on the mechanical optimization, and the functions that they will fulfill the robot.

BullsEye

Jasper van Meer, Hermen de Jong, Gijsbert Thomassen, Bastiaan Vergouw, Daan van der Stelt

Wageningen University, Farm Technology Group, Wageningen, The Netherlands

CHASSIS				SENSORS	
WxLxH (cm):	**50x110x60**	Weight (kg):	**35**	☒ Camera	☒ Laser
Model/Make:	**LT-3.0i**	Number of wheels:	**4**	☒ Compass	☒ Odometry
Drivetrain concept:	**Direct drive**	Turning radius (cm):	**15**	☐ Sonar	☒ Gyroscope
Battery time:	**1h45m**	Rain resistance:	**Splash proof**	☐ IR	☐ Mechanical sensor

Robot software description
The software is build up in LabVIEW

Robot hardware description
Computer: Gigabyte GA-B75N, Intel Core i7 3770T Compass: Xsens Motion Tracker MTi Laser: Sick Laserscanner LMS111-10100 Batteries: ZIPPY Flightmax 8000 mAh 6S1P 30C Cameras: Guppy F033C
Task strategy description
Task 1: The laser scanner scans the rows and with that information we make a preferred route with an intersect and a slope. **Task 2:** Using laser scanner, which recognizes objects. **Task 3:** Using vision (cameras) and comparing colours/shapes. LED lights and a patlite are used for signalization with lights and a speaker for a sound signal. We will save the location of the golf balls in a map by using a GPS module. **Task 4 (Cooperation):** We are not going to prepare this before the event starts, if we find a partner during the event to do this task we will enter this part of the FRE **Task 5 (freestyle):** To spray the golf balls, between the rows and also in the path**.**

Team Ceres

Jelle Adema, Daan van den Brandt, Tobias Davids, Jannick Eichert, Bram Geurts, Stefan Heijnen, Niek Lamers, Simon Norra, Nick Staaks, Patrick Surrey

Fontys FHTenl Venlo, Mechatronics & Software Engineering, Venlo, The Netherlands

1. Introduction

The main objective of this project is to get a smooth and robust system. During the

event, the robot will have to drive smooth and not make wrong decisions (i.e. left turns instead of right, start turning when the robot is still in a row, etc.) also the software should run without getting errors that cause the robot to get stuck in one state.

Before going to the event, the complete system already should be tested in Venlo. All parts of the system should also be tested separately before implementation.

During the event, no parts can break that cause the robot not to function anymore. All maged parts that can break during a next task have to be replaced before the task. All the tools necessary to do this should be available at the event. The reserve parts have to be manufactured in the mechanical lab at Fontys Venlo. Two full sets of batteries have to be available at the event (one in the robot and one as backup).

When a weed plant is detected (golf ball task), the lights have to flash when the robot is standing next to it and the GPS data of that point has to be used.

The robot needs to have an average speed of at least 0.7 m/s. When the robot isn't detecting weed plants the speed for navigating in rows has to be 3m/s. The robot has to drive out of a row within 1.5 seconds and turn into a new row in 2 seconds. The robot should stay within the boundaries given by the Field Robot Event organization.

Implement a user interface to control the robot during the event. The user should be able to select a task, the path the robot has to make (for task 2) and a speed. The user will be able to see the amount of weed plants found and their coordinates and the battery status.

2. Mechanics

For this year, the decision was made to have the mechanical parts of the field robot as a modular system. This means that all of the parts can be replaced separately. The advantage that this gives us, is that we can replace a single part without having to redesign the whole concept.

A combination of multiple gears makes it possible for the robot to drive. A signal is sent from the control box to the dc motors. The dc motors will start to work and this allows the gearbox to do its work, which is making the wheels spin. The direction of the wheels is determined by the position of the gear rack. The position of this gear rack is determined by the stepper motor. This stepper motor gets an input signal from the control box. Feedback regarding the position of the gear rack is given by the position of a variable resistor. This resistor is connected directly to the gear rack.

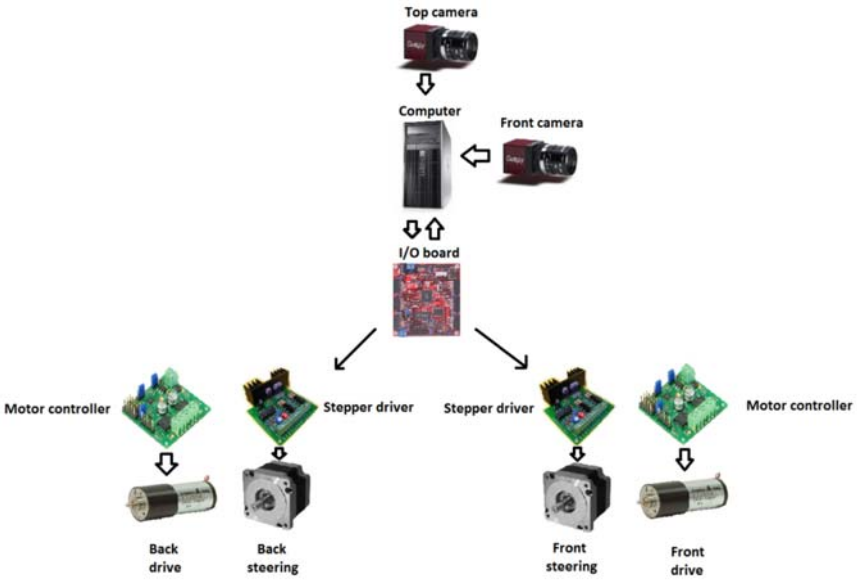
The chassis has been reduced in size in comparison to last year. This makes it possible for the robot to make a smaller turn. The wishbones have been reduced in sized to make this possible. However, this resulted in us having to adjust the torque transmission. We also had to find a replacement for the spring that connected the wishbones to the gearbox.

3. Sensors

The whole robot is based on camera- and compass navigation. Below you can find a description of the Hardware and Software.

3.1 Hardware

In the picture below you can see an overview for the hardware. The top and front camera are connected to the pc through Firewire. The computer sends the serial commands to the I/O board. The stepper drivers and motor controllers get their data from the I/O board. The robot has 4 wheels for driving and the front and back wheels can turn. Therefore there are 2 drive motors and two steering (stepper) motors. 4 LiPo batteries power the whole system with 24V.



Both cameras are made by Allied Vision, a company for industrial vision products. The I/O board is a microcontroller named Cerebot 32MX4. The motor controllers are custom made by our electronics-team and drive two maxon-motors in the front and in the back of the robot. The stepper motors for steering are driven by TB6560 stepper motor drivers.

The drive part is responsible for the drive speed and steering of the robot. Both actions will be mostly decided on information from vision. The programming is split in to separate cases, each cases handles one of the tasks of the event. These cases will be discussed below.

Task 1 “Drive”

For this task we will receive an angle from vision. The angle represents the orientation of the robot in respect to the row. Using this angle this angle the robot will be steered through the rows. At the end of the row vision sends an end row signal. The first time the end row signal is triggered a string input with either R of L will indicate the direction of the first turn. The robot will then preform a turn out row of 90^0. This turn angle will be measured using a compass. Afterwards a "turn in row" will we started off again 90^0, the only difference is that this action can be interrupted as soon as the reference angel to the row is detected.

Task 2 ”Path”

The main drive function from task 1 will be used with some small differences. For the second task a path will has to be used. This path can be put in at the user interface. When the robot encounters an end row signal it will extract the information for that turn from the input string. This decides de direction of the turn and the number of rows to skip.

Task 3 “weed detection”

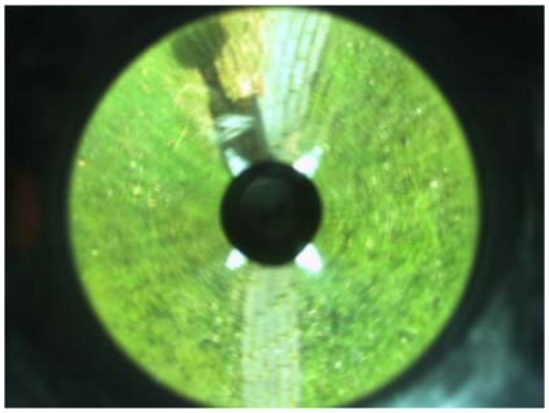
Much like the last task, this one will also use the basic functionality’s of task 1. The difference for this task is the weed detection. When vision detects a weed (golf ball) they will send a signal if the weed is either at the left or the right site of the robot. When receiving this signal the robot will stop and blink the lights at the side of the robot where the weed is detected and the GPS data will be saved.

The only human input to the robot will come from the industrial tablet IO-10 from Lextron. The tablet has Windows 7 Professional as operating system and not Android which is popular among most normal tablets. Windows owns most of the market share within the industrial tablets and it can be easily combined with the PC which is controlling the robot. It will be used to select the task the robot has to perform after the start button has been pressed. The tablet is water and dustproof (IP65) and thus it will be a good tablet for in the field. Also the screen is still readable in the sunlight and even if you’re wearing gloves the tablet is still usable. The robot will work fully

autonomously then and will only give feedback to the tablet. The other option with the tablet is to control the robot manually. The robot can then be controlled with the arrow keys for the direction and the speed has to be set on a constant value. With the tablet you can also check both camera images while the robot is driving through the field performing ist task.

3.2 Software and strategy

Our field robot uses vision to navigate through the lines of maize plants. A top camera and a front camera are connected with the computer via Firewire. The pictures below show the two camera images.



The top camera looks up against a mirror. With the use of this mirror we can look around 360°. We use the top camera for driving and general navigation.

The front camera looks forward and is used for detecting cones and finding the golf balls. Further we use the front camera for perpendicular driving.

Both camera's calculate a driving angle. These angles are compared in the software and a decision is made which camera or what value is send to the wheels.

The vision software runs, as same as the drive software, on Labview. The image processing is done in 5 separate while loops. So all programs execute independently from each other. The cameras do acquisition with 50 fps. This is the limiting factor rom the software now.

The row detection is the main part of the programm and will be needed in every task to guide the robot through the rows.

The field robot gets the image from the top camera as an input and creates the steering angles and the signal for the end of a row out of this. This progress work as follows:

First a mask will be applied on the original image so that all unnecessary parts of the image will be removed. In a second step a single colour plane will be extracted on which a threshold will be applied. Afterwards some small particles will be removed and little holes will be filled. At the end the labview program tries to detect angles out of the processed image, to send them to the driving part of the program. If no angles will be detected, the robot is at the end of a row.

The row counting program will be needed in the second tasks to count the rows while driving perpendicular to them. This is necessary to turn into the correct row like it's given in the program. The process is the nearly the same as in the row detection program:

First a mask will be put on the original image, depending on the field orientation. In a second step a single colour plane will be extracted. Then a threshold will be applied on the image, particles will be removed and holes will be filled. When you now drive perpendicular to the rows, they will pass through a region of interest and will be counted. The number of counted rows will be given to the drive part of the labview program so that the robot turns in the row when he counts the correct number.

4. Conclusion

We build a field robot, which can guide autonomous through a maize field with just using camera and compass information. He is able to drive up to 3 m/s and to steer within a very small turn radius. The camera-images are processed very fast because of several while loops in the software, which are running parallel. We hope that this situation will be the same during the upcoming Field Robot Event.

Cornstar robot

Miran LAKOTA[1], Peter BERK[1], Jurij RAKUN[1], Peter LEPEJ[2], Jože KRANER[2], Miran PURGAJ[1], Zvonko Šlamberger[1], Uroš JAGODIČ[3]

[1]*Faculty of Agriculture and Life Sciences, University of Maribor, Hoce, Slovenia*
[2]*Faculty of Electrical Engineering and Computer Science, University of Maribor, Maribor, Slovenia*
[3]*Faculty of Natural Sciences and Mathematics, Maribor, Slovenia*

1. Introduction

In the age of technological revolution agriculture is one of the disciplines that have a bright future. As big food producers rely on the use of heavy machinery, this is still not the case for mid- and small-sized farms, which can pose a potential food safety problem. If handled manually, food can transmit disease from person to person as well as serve as a growth medium for potentially harmful bacteria. Nevertheless, some work still demands manual labour that is time consuming, exhausting and expensive. The thought of introducing a small army of intelligent robots to do the job quicker and more accurate seems appealing, but we are not just there yet. For one, natural uncontrolled environment poses a challenge with its changing conditions. An overview on the subject showed that there are some potentially good solutions but the authors rely on specific conditions (like night time) or their solution is designed to work in controlled environments (green house) and some are simply too big or too heavy to be useful at this stage. In this paper we try to tackle the problem by introducing our own mobile agricultural platform.

In order to achieve our goal, we decided to put our efforts to build a small autonomous self oriented robot, that could for instance serve as a potential tool for selective pesticide spraying, fertilizer applicator or even as a device that could estimate the yield at the end of the harvest by simply taking digitalized snapshots of the tree canopies. In the following section we start by describing our solution in detail.

2. Mechanics

The robot consists of four major components: mechanics, actuators, embedded electronics and sensors. The mechanical part includes a four-wheel drive, two axles that can be turned individual by servomotors, automatic gearbox with forward and backward two-speed trasmittion and a three-phase motor. The actuator part includes a pressured reservoir, two electro-magnetic valves and nozzles that enable the robot to spray. The onboard digital camera and a laser range sensor make up the sensoric part of the robot. The last but not least is the electronic part of the robot, which includes an embedded computer, peripheral circuit and a power supply. The render

image of the robot is depicted on Fig. 1, showing all crucial components of the robot, while Fig. 2 depicts the robot in action while spraying roses.

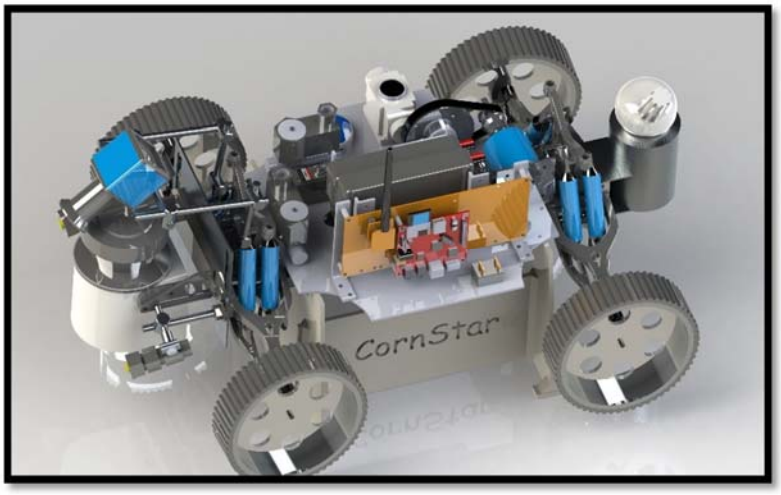


Fig. 1. Render of a Constar robot – depicting all its crucial components.

Fig. 2. Mobile robot while spraying roses.

3. Sensors

The sensors on the robot include an industrial camera (The imaging source's DBK31BU03) and a laser range scanner (SICK TIM 310). The camera captures Bayer

encoded images with resolution of 1024×768 pixels at 30 fps. The camera is connected to the embedded computer by using an USB connection. The laser range sensor captures distances between a sensor and an obstacle in a 270º radius from 0.05 up to 4 m distance. The sampling frequency of the laser scanner is 15 Hz with 1º resolution. It is connected to the embedded computer via USB connection

3.1 Hardware

The electronic part of the robot is made up of an embedded computer (Nitrogen6x) and a peripheral expansion board build around a AVR ATmega 128 microcontroller. The computer is based on AVR Cortex A9 quad core processor running at 1 GHz, with 1 GB of memory and can reach a performance of 2000 BogoMIPS per core. It offers a vast number of ports, where USB was used for camera and laser range sensors, while the UART port is used to communicate with the expansion board. An Ubuntu Linux (Linaro) distribution was selected for the operating system that was uploaded to a SD card, where a custom version of the kernel had to be compiled to support the USB based camera.

The camera, the laser range sensor and the embedded circuit are all powered by two 3 cell LiPo batteries connected in parallel. They can provide a constant 175 A or 350 A peak current (each), 11.1V voltage and 3800 mAh capacity (each). In order to power the circuit the voltage is lowered to TTL voltage levels using a switching regulator.

3.2 Software and strategy

The embedded computer runs on a Linaro Linux operating system where a custom build version of the kernel was compiled in order to support the camera. In addition, in order to control low-level hardware the mobile robot uses a Robotic Operating System (ROS), which is a meta operating system. ROS provides a number of libraries and tools for different robotic applications; it includes hardware drivers, device drivers, visualizers, messages passing and package management, all this as open source software. ROS supports code reuse in robotics research and development, it is a distributed framework of processes where processes can be grouped in packages. For the mobile robot we used device drivers for Sick laser scanner and other hardware, provided by ROS. In order to connect the embedded computer with the mobile base we created own program for command interpretation and communication protocol.

The navigation algorithm relies on the distance measurements from the LIDAR sensor. It detects obstacles up to 4 m away and then decides what is the optimal path the robot should take. Once the robot reaches the end of the crop line it uses data from the on-board compass and turns in the next row.

The known objects the robot is familiar with are detected with the help of the digital camera. It uses a colour segmentation technique to separate the background from the known objects. Once the objects are detected the robot then sprays them or just save the location of their position.

4. Conclusion

With the fifth generation of the Cornstar field robot we are a step closer to a prototype robot that could serve as a small agricultural tool for various tasks. Of course this is only the first step toward building a universal tool that could do the everyday work quicker, more precise and without human interaction. We are satisfied with the current results, but still have a way to go.

The main problem we faced this year was the accuracy of the robot movement where the mechanical part should be replaced with a precise one. Additional sensors such as odometry or IMU would also be welcome and are to be used next year. This way we could achieve a more settle and accurate movement of the robot.

5. References

R. C. Gonzales, R. E. Woods, 2001. *Digital Image Processing*, 3rd edition, Upper Saddle River: Prentice Hall PTR.

J. F. Thompson, J. V. Stafford, P. C. H. Miller, *Potential for automatic weed detection and selective herbicide application*, Crop Protection, vol. 10, pp. 254-259, 1991.

L. Shapiro, G. Stockman, *Computer Vision*, Prentice-Hall, Inc. 2001.

Bulanon, D.M., T. Kataoka, Y. Ota, and T. Hiroma. *A Machine Vision System for the Apple Harvesting Robot*, Agricultural Engineering International: the CIGR Journal of Scientific Research and Development. Manuscript PM 01 006. Vol. III.

Guo Feng, Cao Qixin, Nagata Masateru, *Fruit Detachment and Classification Method for Strawberry Harvesting Robot*, International Journal of Advanced Robotic Systems, vol. 5, no. 1, 2008.

Grisetti, G., Stachniss, C., Burgard, W., Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters, IEEE Transactions in Robotics, Volume 23, pp. 34-46, (2007)

Grisetti, G., Stachniss, C., Burgard, W., Improving Grid-based SLAM with Rao Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling, Proceedings of the IEEE International Conference on Robotics and Automation, (2005)

Kohlbrecher, S., Meyer, J., Peterson, K., Graber, T., Hector SLAM for robust mapping in USAR environments, ROS RoboCup Rescue Summer School Graz, (2012)

Kohlbrecher, S., Meyer, J., von Stryk, O., Klingauf, U., A Flexsible and Scalable SLAM System with Full 3D Motion Estimation, Proceedings of the 2011 IEEE International Symposium on Safety, Security and Rescue Robotics, Japan (2011)

Stone, H. S., A Fast Direct Fourier-Based Algorithm for Subpixel Registration of Images, IEEE Transactions on Geoscience and Remote Sensing, Volume 39, Number 10, pp. 2235-2242, (2001)

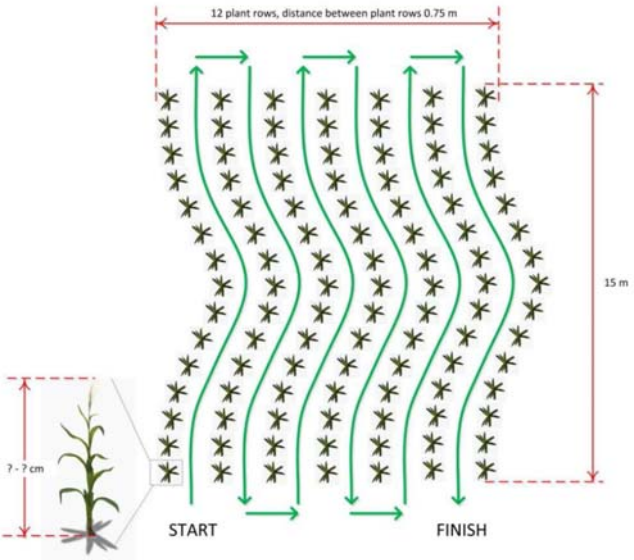
DTU Maize Monster

Morten Nylin, Mikkel Magdal Mortensen, Christian Myrhøj, Jan Lorenz Svensen, Ole Ravn, Nils Axel Andersen

Technical University of Denmark (DTU), Electrical Engineering, Lyngby, Denmark

1. Introduction

Field Robot Event is an international robot competition with the purpose of promoting agricultural robots. This year it was held in Strenzfeld, Germany with The University of Hohenheim as hosts. The competition consists of 5 tasks where the first 3 are part of the championship and the 2 last is optional.

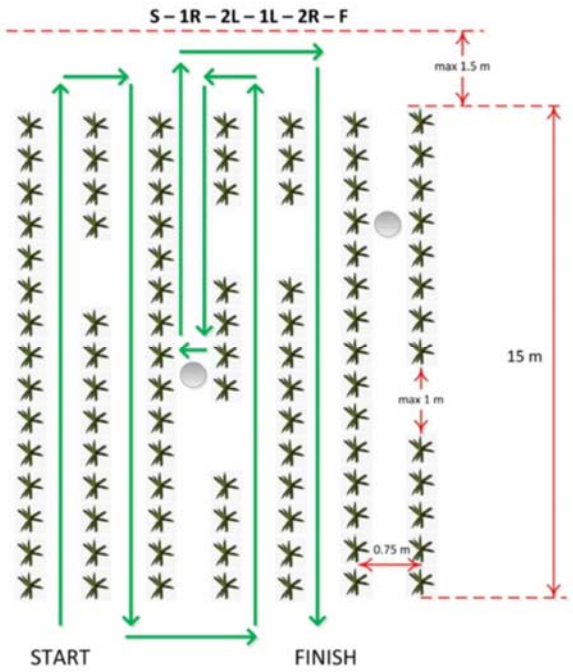


Task 1 Basic navigation in curved rows of maize

Task 1: Basic Navigation

The robot has three minutes to cover as much distance as possible navigating in long curved rows of maize plants

Task 2: Advanced Navigation



Task 2: Advanced navigation in straight rows of maize

The robot has five minutes to navigate through straight rows of maize following a predefined path. Rows can be blocked and plants can be missing in both sides with a maximum length of 1 m.

Task 3: Professional application

The robot has five minutes to identify five golf balls placed randomly along curved rows of maize. When a golf ball is detected a sound or light signal is given and the position in GPS coordinates should be logged within 0.75 m precision.

Task 4: Cooperation

Two robots have to perform a task cooperatively. The task is chosen freely, but there has to be communication between the robots.

Task 5: Freestyle

The robot can do a freestyle task of own choice with creativity, fun and application oriented performance in mind.

This year DTU participated in the task 1 to 3.

DTU Maize Monster anno 2014

2. Mechanics

The robot dimensions are L x W x H = 82 cm x 37 cm x 47 cm and the weight is around 20 kg. The back wheels are two 12 V brushless electro motors driven with differential drive. The servos used for the front wheel steering is a pair of Dynamixel MX-64, thus making the overall steering an Ackerman steering. The robot is tested with speeds up til 1.7 m/s, but is stable at 1.0 m/s. The front wheels are attached like on a tractor to get a better grip on the ground when driving in uneven terrain. The robot is powered by two 12 V batteries connected in parallel located under the robot between the wheels. The wheels are 8 cm in radius.

3. Sensors

For vision two Kinects are used, and for navigation two Hokuyo URG-04LX laser-scanners. Two encoders mounted on the back wheels are used for odometry, and during task 3 a GPS module was connected as well.

3.1 Hardware

The motherboard is a ZOTAC ION, the standard type used for Small Mobile Robots (SMR) at DTU Automation and Control. For the motor control a Teensy++ 2.0 is used.

3.2 Software and strategy

The on board computer is running a Linux kernel compiled specifically for the project. Besides this, the robot is running DTU Mobotware, which is developed and maintained by the DTU Automation and Control. The communication structure of DTU Mobotware is illustrated below.

The real-time hardware daemon (RHD) handles data communication from sensors and makes it available for the Mobile Robot Control (MRC). The `ulmsserver` runs the laser scanners and the `ucamsserver` runs the Kinects. All the data is available for easy access in the MRC which can be controlled by scripts written in SMR-CL. This is an advantage as it is possible to debug and test without the need for compiling code. It is possible to write plugins for the MRC, for specialized operations.

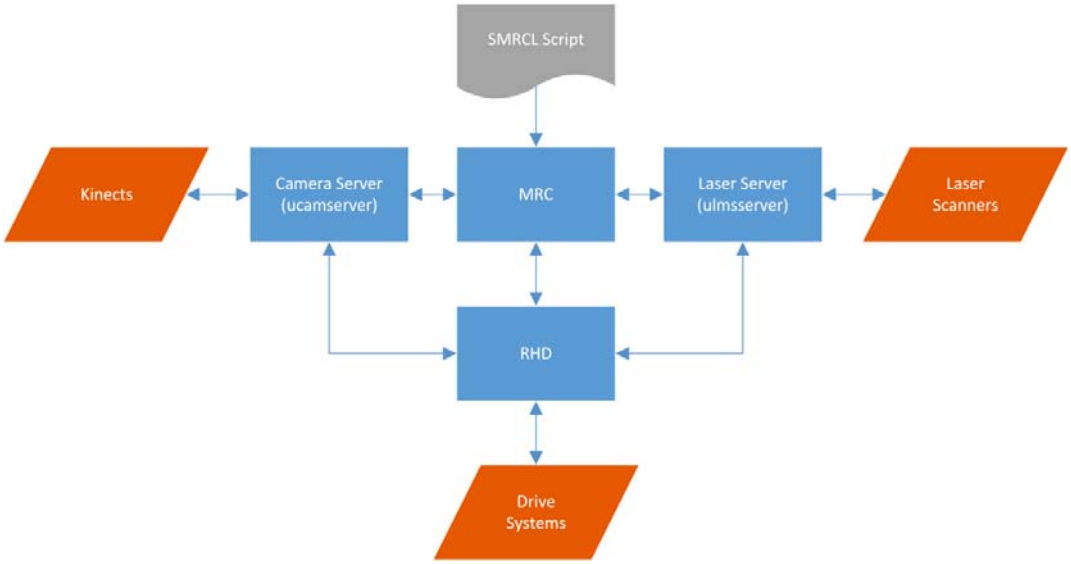


Figure 1.1: DTU Mobotware communication structure

Drive Algorithm

The robots driving algorithms is the based on the same basic design in all three tasks. The basic design is the algorithm found in task 1, where task 2 and task 3 has some modifications and additions.

Task 1: Basic design

The basic design is made up of three parts: The control-, the drive- and the turn part.

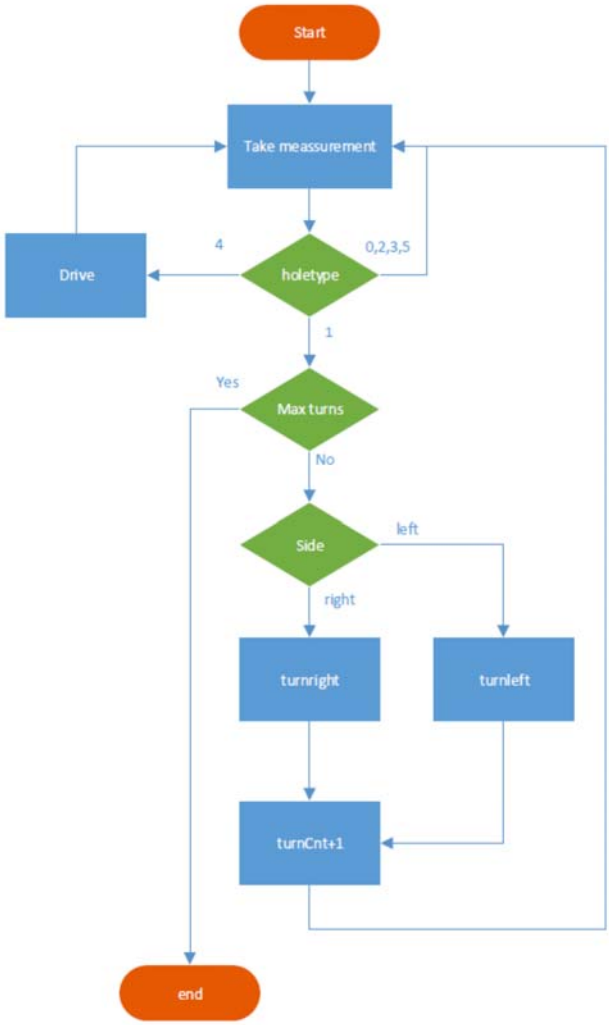
The control part handles whether the robot should turn, drive or stop the program. The decision is based on the feedback from calls to the Rosebot (FRE 2012 winner) [3] plug-in, described later, by evaluating which kind of hole type is in front of the robot. If the Type is 0, 2, 3, 5: Take new measurement, if Type is 1: Initiate adjacent turn process, or end the run if max turns is taken. If Type is 4: Drive accordingly to the feedback x,y and theta given by the plug-in.

The robot drives 3 cm before returning to the control part. The robot is driven with the command "`driveon`". While the robot runs the drive part, it keep tabs on the distance driven in the current row, if a predefined distance is reached, the speed is set to a predefined slow speed.

In the turn part the robot turns ±90 degrees with a radius of 0.5 m, followed by backing 1.1 m. In order to find the right row to go down through, the robot uses the function `FindRows` in the Rosebot plug-in, to find the direction and coordinate of the closets point on a line corresponding to nearest row in front of it.

If the distance to this line is smaller than 0.15 m or larger than 1.5 m, the robot will back up 0.1 m and take new a measurement, repeating this until the distance is accepted. Then it will drive to the found line. If the robot finds no rows, the robot will end the run.

The robot then measures the distance to the next row, and determines the needed turn radius and turn angle, if the radius is less than 0.4m or larger than 0.8m, it drives forward 0.1m and takes another measurement. If the turn radius is accepted the robot will turn accordingly to the parameters it determined, and then return to the control part.



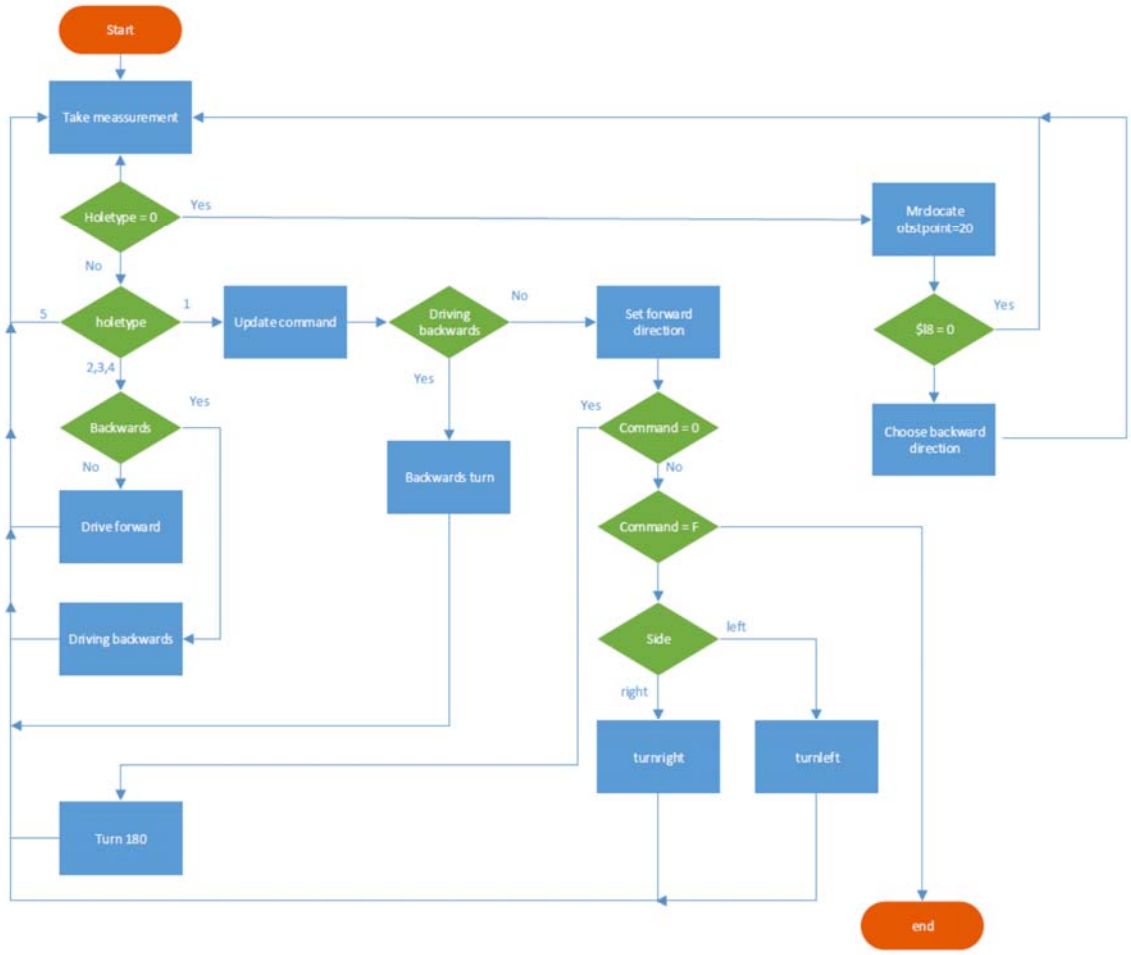
Flow chart of the basic drive algorithm

Changes for Task 2

The task 2 version has changes in all parts and an added obstacle part. In the control part a predefined drive plan is introduced, this drive plan is used to decide the robots next move, turn left, turn right etc., and is used when a hole type 1 is found.

When the control part returns a hole type 0, the robot will run the `obstcheck` plug-in, for testing if there really is an obstacle or if just some noise or leaves. If an obstacle is found, the robot will drive backwards through the row to the end.

While the robot is backing the speed limitation described above is ignored. If it has backed the same distance it drove forward through the current row or registers a hole type 1, the robot will go to check the drive plan for the next step.



Flow chart of task 2 drive algorithm

The turn part is expanded with a loop, so if the robot has found a row and the plan is to drive down a later row. The robot will then continue to drive to the found row and search for the next row, until the right number of rows is passed.

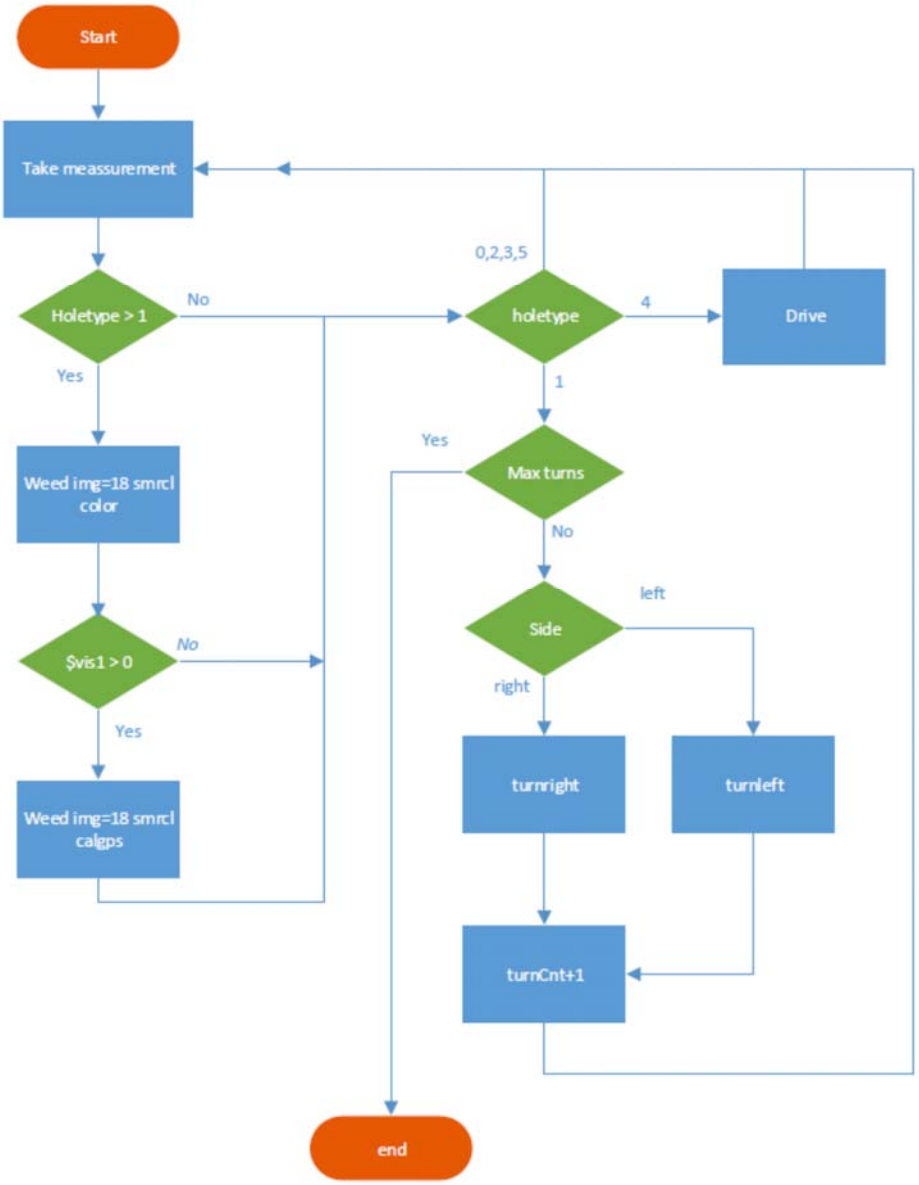
If the drive plan dictates that the robot should return down the same row it just came from, the robot will drive out of the row with a turn, back out and drive in the row again with a new turn.

If the robot backed out of the row, then the first part of the turn algorithm is changed. For left and right turns the turning amount is reversed, up till the first measurement is taken, where after the proceedings are normal again. If it should drive down the same row it will ignore the turn and just drive straight ahead,

Changes for Task 3

The turn part is the same as used for task 2, with the robot driving down every second row. The main difference is that an imaging part has been added. Each time the control part is entered, the imaging part will run. When running the robot will take a picture,

and check if there is a possibility for a ball is present, by evaluating the amount of yellow pixels in the picture. This is done using the Weed plug-in. If there is a possibility for a ball, the robot will stop and check once again, for conforming the existence and location of the ball or de-confirm it. It will then return to the control part to evaluate the hole type. If a ball is found the robot will drive a calculate distance before looking for balls again. The distance equals the needed distance to lose sight of the found ball.



Flow chart of task 3 drive algorithm

Rosebot plugin

When utilising the Rosebot plug-in, one passes commands defining which function is to be used: `findRoseRows` or standard

The different forms of holes, the robot can difference between is:

type 0: dead end

type 1: open field

type 2: left wall

type 3: right wall

type 4: hole

type 0: ignored hole

standard command: ""

First the algorithm sorts out all data from the laser scans, such that only points in a certain interval between SCANMIN < r < SCANMAX, where r is the distance to the observed point, and the rest is put to -1. It then passes the data through the `findbesthole` function.

The `findbesthole` function utilises the `findhole` function to find the hole type, and then evaluates the type, if a hole is found its edges and size is evaluated, then `findhole` is called again but the start point in the data is moved to the currently known left edge, the function ends when all data points have been evaluated.

The `findhole` function determines the hole type by looking at the first laser point in the data, if this is outside the interval, the function will look for an open field, and if it finds points in the data with r>0, it will then have found a left wall. If the first point have r>0, then robot will check the data to find the first possible hole, r < 0. If the result is no hole at all it will return hole type 0, else it will search for the next edge; if no edge is found, it returns type 3(right wall) else returns type 4(hole).

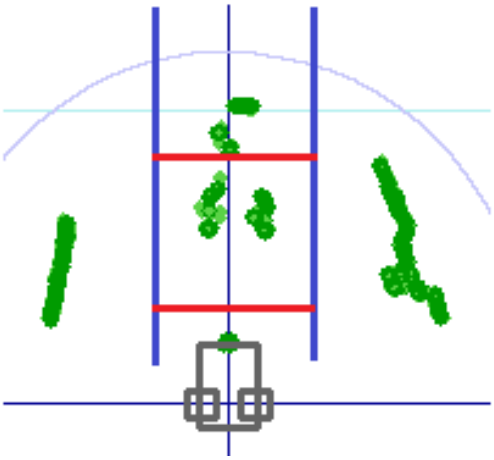
If the resulting hole type returned by `findbesthole` is different from 0 or 1, it is evaluated with regards to the limits of hole size and angle to the hole.

findRoseRows command: "findrow"

By utilising RANSAC algorithm on the laser data, the amount of rows ahead of the robot is found, together with the distance and angle to them. The closets row with an angle in between the accepted interval.

Obstcheck

The following is a description of the plugin `auobstcheck`. The way `auobstcheck` works is relatively simple. It starts out by transforming the laser scanner data from polar to Cartesian. Then it checks whether these points are within a window of interest and if so assigns them to a new array.



Window of interest

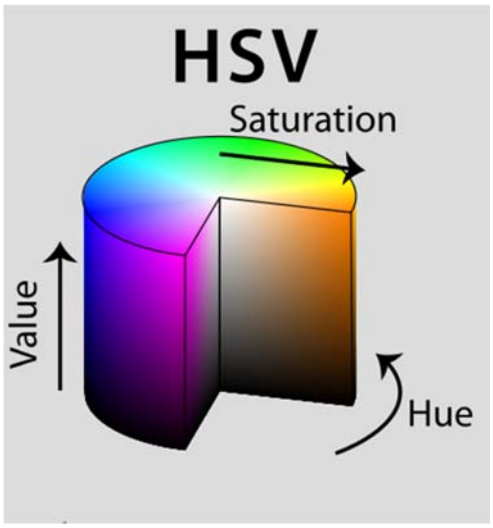
Next the distance between each of the newly assigned points is calculated, starting with the left most point as the first origin for the distance calculation. After the distance from this point to all other points is calculated, the origin is set to the second left most and so on.

If the distance is found to be below 0.15 m. the two points are considered to be part of the same cluster. Only clusters of a minimum size, which can be set externally by the user, are accepted as a valid obstacle.

If the plugin finds an obstacle it will return the value true.

Weed Algorithm

In this section the algorithm for detection of weed will be described. The task of detecting weeds was changed slightly from last year. The weeds was now represented by yellow golf balls and placed on the ground. With this in mind we took the already existing "ball-finding" plug-in and modified it to find the right color and changed the masking.



To detect the yellow color of the ball we chose to convert the RGB image to HSV, which is Hue, Saturation, and Value.

The HSV color model can be described as a cylinder with hue being the angle, saturation the radius, and value as the height in the cylinder. The specific color is the hue, where the amount of color is the saturation, and then the amount of white mixed into the color is the value. This can be seen in the figure above.

The reason for changing the color model is the problem of lighting. When using the RGB color model a pixel detected would in very poor lighting have lower values than if the pixel was illuminated. This is due to structure of the model, where RGB is a cube having three coordinates changing with illumination. By using HSV it is apparent from the figure that in theory only the height coordinate change and therefore not changing the hue / saturation relationship.

This means that with the right hue and saturation a color can be detected no matter the lighting. This is although only in theory, and instead of a specific hue and saturation a range for these two was chosen.

The color model then allowed us to specific threshold the pixels of the right color

4. Conclusion

With the drive algorithm described above, the robot showed robust and very stable performance during practice, both when driving between the maize plants and when turning in the headland. The robustness on the other hand resulted in slow driving speeds compared to fastest of the competitors.

When testing task 2 the `obstcheck` algorithm turned out to be unstable if the robot did not point directly at the obstacle or if knocked down plants would hang in front of the robot. This is a part that would need improvement before next year's competition.

The ball detection algorithm in task 3 showed some sensitivity to sunlight, meaning an increase in the number of false balls detected. None the less it was able to detect balls without the both with an added shadow and without.

In the actual competition we failed drive in the first task and only received 1 point in the second due to a hardware failure. In the third one we got 4 points leaving us with a total of 5 points and an overall rank of 9th place.

Using the DTU Mobotware software the robot has a great strength, because it allows the robot to be easily reprogrammed and optimized, without compiling huge amounts of code.

The Field Robot Event 2014 was well organized, and without accidents in FRE2015 the robot should be able to get in top 3 or top 5 depending on the competition.

5. References

[1] Siegwart, Nourbakhsh, Scaramuzza, Introduction to Autonomous Mobile Robots, 2nd edition

[2] Beck, A. B., Andersen, N. A., Andersen, J. C., & Ravn, O. (2010). Mobotware – A Plug-in Based Framework for Mobile Robots. In IAV 2010. International Federation of Automatic Control.

[3] Field Robot Event 2012 - task 1, Rosebot winner of the championship http://youtu.be/qF6wc2ls4kk

Eduro Maxi HD

Milan Kroulik, Martin Dlouhy, Stanislav Petrasek, Josef Pavlicek, Petr Hanzlik

Czech University of Life Sciences (CULS), Agricultural Machines, Prag, Czech Republic

1. Introduction

Eduro Team participated on Field Robot Event several times already already ([1]]) so for year 2014 we planned to present something new and use FRE as demo/show opportunity. We planned to have walking robot (FireAnt) AND/OR flying robot (Heidi - Parrot AR Drone2) to do standard FRE tasks 1, 2 and 3. We changed our mind But shortly before the contest due to the updated requirements. It was mandatory to carry heavy (more than 1kg) GPS receiver in the Professional Task and the robot for all tasks 1, 2 and 3 had to be the same. Our fall-back was "Eduro Maxi HD", proved platform from 2010, 2012 and 2013 competitions. Never-the-less there was still space left to present FireAnt and Heidi in tasks 4 and 5.

2. Mechanics

Eduro Maxi HD is the prototype of three-wheel outdoor robot with a differential drive. It is a modular robotic platform for education, research and contests. It weights about 15 kg with the dimensions of 38x60x56 cm. HD version uses SMAC (Stepper Motor - Adaptive Control) drives with belt transmission. The power supply is managed by two Pb batteries 12V/8Ah. The brain of the robot is single board computer with AMD Geode CPU, 256 MB RAM, compact flash card, wi-fi, 3 Ethernet, 1 RS232 and 2 USB ports. The RS232 port is used for connection to CAN bus via RS232-CAN converter. The most of sensors, actuators and other modules (display, beeper, power management etc.) are connected to CAN bus, which forms the backbone of the robot. The more data rate demanding sensors are connected directly to PC via ethernet interface. Two main sensors were used for the Field Robot Event. The weed detection was provided by an IP camera with fish-eye lens. For obstacle detection, the laser range finder SICK LMS100 was used. The robot is further equipped with sonar, GPS and compass, but these sensors were not used in main tasks during the Field Robot Event.

Hexapod FireAnt and quadcoper Parrot AR Drone2 are commercially available platforms. FireAnt is a construction kit from Orion Robotics ([2]). It has 25 force-feedback servos and they are controlled by Arduino board combined with Servo shield.

Parrot AR Drone 2 ([3]) is very successful platform of remotely controlled quadcopter via WiFi connection. The manual control can be replace by autonomous behavior realized by standard notebook.

3. Sensors

3.1 Hardware

The main sensor on the Eduro platform is laser range finder SICK LMS-100. The scanning plane is at 30cm above the ground and it is slightly tilted so at distance 2 meters it sees the ground (on a flat surface). There was an alternative navigation solution, in the case of very low plants, using IP camera from Arecont Vision equipped with wide angle lenses. Finally Eduro has very good odometry and thanks to 3-wheel construction it is useful even in outdoor field terrain.

The Eduro is controlled via CAN bus accessible by bridge to RS232. The main computer is older router running embedded Linux OS. Note, that Eduro has more sensors (magnetometer, sonar) which were collected data but not used in algorithm.

FireAnt had only touch/force sensors this year. Each servo reports its own position and current applied force at 25Hz rate. The development was temporarily paused but you will hopefully see it on FRE2015 again.

Heidi (our Parrot AR Drone 2) robot platform is surprisingly rich of sensors (taken into account price around $300). It has two cameras (front and bottom), 3D IMU (accelerometers, gyroscopes and magnetometer), sonar for height measurement, temperature and pressure sensor. The quadcopter uses 1000mAh, 11.1V LiPo batteries.

3.2 Software and strategy

The software is written in Python programming language and is freely available on GitHub web site ([4]). The Eduro code is very similar to the old code from 2010. The main changes were in improved direction estimation (Advanced Task with holes in plants rows), navigation at the end of rows and camera and GPS processing in Professional Task.

The primary navigation sensor was laser range finder. Array of 541 readings, corresponding to 270 degrees range, where first grouped by 5 degrees. In every group shortest reading was used (rejected were 0 readings = no reflection, or strong sunlight) to represent given angle. Finally experimentally approved threshold 1 meter was applied for decision if is in given angle obstacle/plant.

The algorithm task was to find a gap of given size. The processing was almost context free except variable containing previous position of found center. New direction was set in the middle of the gap. If the gap was too wide (missing plans or end of field) then alternative strategy was used (see below).

During testing we find out that the data from odometry/encoders are superior to compass/magenetometer readings. Because of that we deleted code using compass azimuth as reference and replaced it by "dynamic reference point" which was delayed by 2 seconds (see *self.poseHistory* in *fre.py* code). This was mainly used in cases when gap did not have desired width or where more then one option was available.

The second major change was navigation between rows at headland. The idea was to "follow a wall" on the left or right side (depending on the first turn) and count row openings. This could work for long transitions but was not reliable due to the uncertainty of robot initial position. Also the plans were hardly visible on the robot side (they were smaller than 30cm, at least on semi-damaged testing field). So at the end we used our former code with odometry only and better calibrated transition to 2nd and 3rd row.

Quite a challenge was integration of RTK GPS receiver. It is very difficult to get it working, parse data, integrate it to the main code, and test it all in 20 minutes! But it can be done . It is true that we had code tested with standard GPS and already verified USB-serial converter. We managed to do two tests in the field where the first onw was just to log data and the second run was already integration to the system.

We should also mention code for yellow balls recognition. Yellow is not simply distinguishable color (you can see it on dry leaves, for example). We classified each pixel separately and counted number of yellow pixels, its mass point and variance. While other teams modified their robots due to strong sunlight we did changes only in software. Basically instead of yellow we were looking for white (well, we did not realize that at start position we will see white start line). We set all filters to extremes to reduce the false detection - yellow was detected only in a small region, there was both lower and upper limit for number of yellow pixels, and they had to be very very bright.

4. Conclusion

Eduro Maxi HD was not the fastest robot but it was one of the most precise and most reliable. With a bit of luck this was enough to score well in all main FRE2014 tasks: it reached 2nd place in Basic Task, 2nd place in Advanced Task, and 1st place in Professional Task. As consequence we won, together with Phaeton team, the main FRE2014 contest.

5. References

[1] http://robotika.cz/competitions/fieldrobot/
[2] http://www.orionrobotics.com/FireAnt-Hexapod-Robot_p_248.html
[3] http://ardrone2.parrot.com/
[4] https://github.com/robotika/eduro

FloriBot

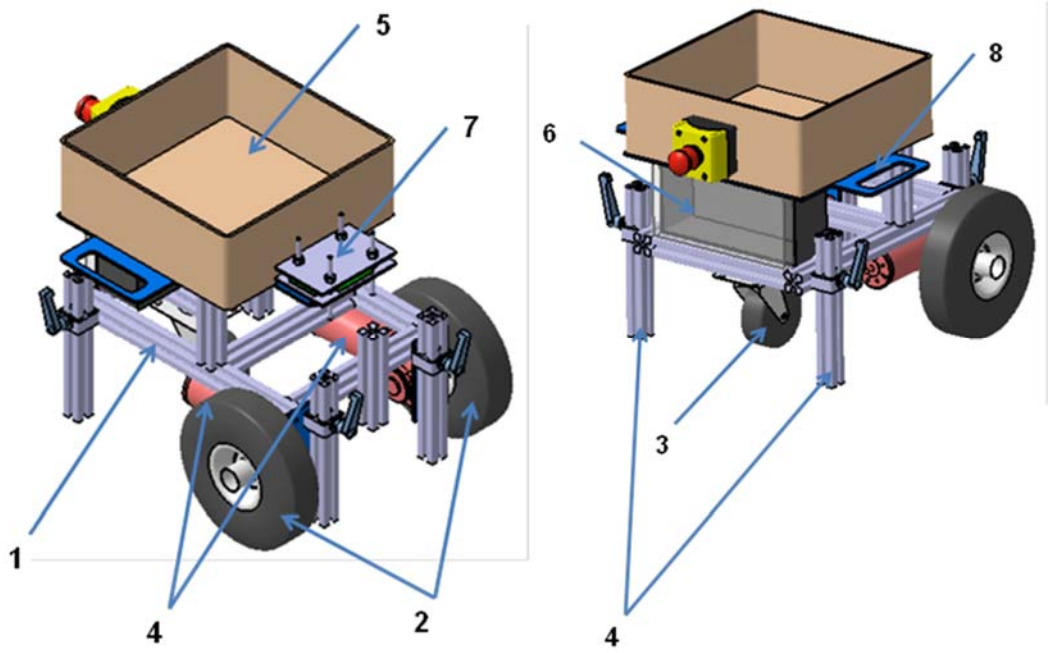
Björn Eistel, Dominik Teltscher, Felix Binder and Manuel Rixen

Heilbronn University of Applied Sciences, Mechanics and Electrics Electronics, Heilbronn, Germany

1. Technical Concept

The hardware of the mobile robot FloriBot can be divided into the sectors: chassis, undercarriage, power train, body housing and periphery.

FloriBot's chassis is the fundamental bearing structure of the mobile robot. It is composed of standard aluminium profiles (1) whose square cross-section has an edge length of 30 millimeters. Because of these multifunctional components the chassis provides a flexible basis for future chassis since the profiles can be cut to length or replaced very easily.



The chassis is carried by the undercarriage which consists of two separately driven front wheels (2) and one free spinning castor wheel (3) which is pivot-mounted. Due to this constellation of the wheels the undercarriage guarantees low-frictional cornering.

Each front wheel is driven by one DC electric motor (4) with a rated speed of 82 revolutions per minute and 3.46 Newton meters of torque. With these motors FloriBot reaches a maximum speed of 0.95 meters per second and is able to get over ascents up to five percent. The DC electric motors are connected with worm drives which have a transmission ratio of 1:20.

The body housing is divided in two parts. One splash-proof casing of polycarbonate providing the space for the control unit and CPU (5) and another splash-proof

polycarbonate casing containing the twelve volts accumulator (6) which supplies the energy for the whole FloriBot. Every casing has a removable top which is fixated with four quickly releasable plastic screws. The casings are arranged in a way so that every casing can be opened separately without removing the other one first. Therefore the accumulator can be replaced very fast.

There is a multifunctional adapter plate (7) at the front of the robot. This plate is for carrying the optional periphery like a camera, a pan-tilt-unit or a RFID aerial.

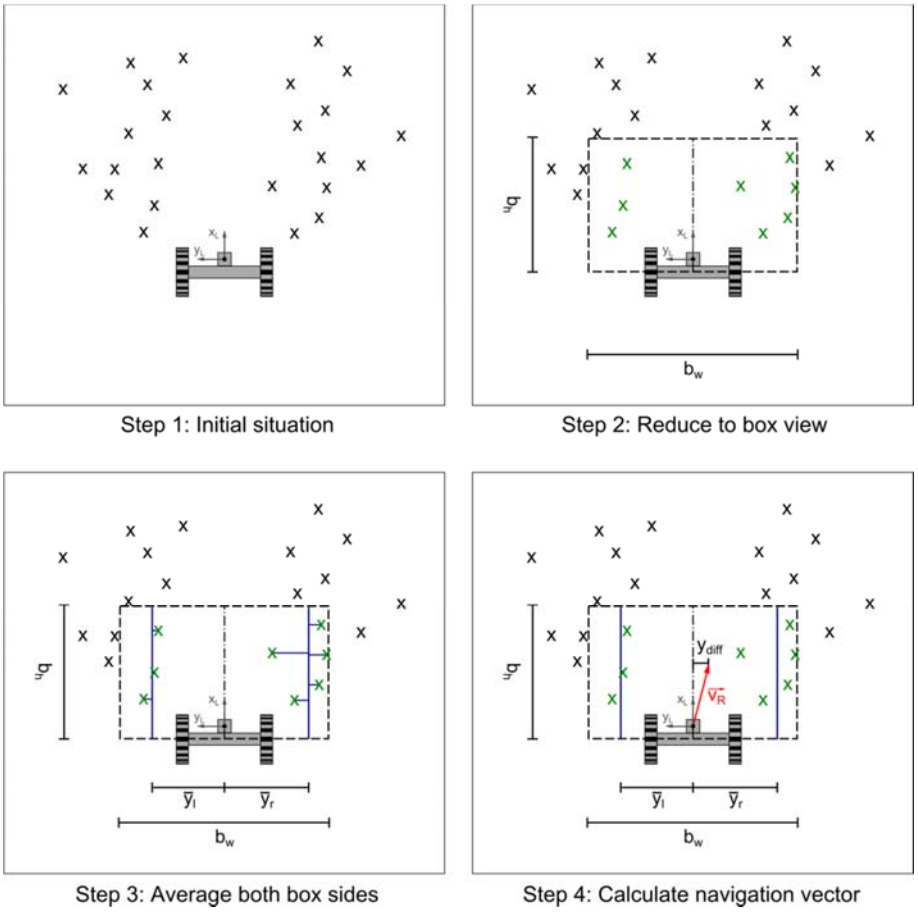
Moreover the FloriBot has carry handles (8) on each side which are coated with foam material. So the robot can be transported comfortably by two persons to the event's start point without unnecessarily losing energy from its battery.

Besides FloriBot has a quick system for jacking up the robot if repair or service is needed. This system consists of four extendible telescope profiles (9) with quick release which bear the robot in serving position.

Altogether FloriBot has an off-size of 484 x 413 x 498 millimeters (length x width x height).

2. Abstract of navigation method

In the following the navigation method of the FloriBot will be presented. The method refers to the commonly used 'potential field method', with some additional simplifications that were made especially for the conditions of the Field Robot Event. It is the goal to develop a principle that helps to find the gap between two lines of random obstacles. The step by step instructions are shown in the following pictures and are explained underneath.



Step 1: Initial situation

First of all we place our robot into a field of obstacles. The obstacles are randomly placed and have an unknown pathway. The only known facts are that the obstacles form two lanes with a more or less constant distance between them. The robot is placed in front of these lanes.

Step 2: Reduce to box view

The first step of this method is to reduce the view of the robot to a small box. This is necessary, because the laser reaches obstacles within 30 m range and the closer an obstacle is, the more important it is. The boxes dimension is determined by the width b_w and the height b_h. Both parameters will be used later to adjust this method to a real maze. Additionally the box is separated into two equal sides next to the center of the coordinate system of our laser sensor.

Step 3: Average both box sides

Now the y-coordinates of every obstacle that are within the borders of the box sides are calculated to an average. The averages y_l and y_r represent an imaginary line of all the obstacles on its side.

Step 4: Calculate navigation vector

With the calculated averages it is now possible to subtract them and get the difference y_{diff}. This difference indicates the final navigation vector

$$\overrightarrow{V_R} = \begin{pmatrix} \frac{b_h}{2} \\ y_{diff} \end{pmatrix}$$

and hence our direction through the lanes.

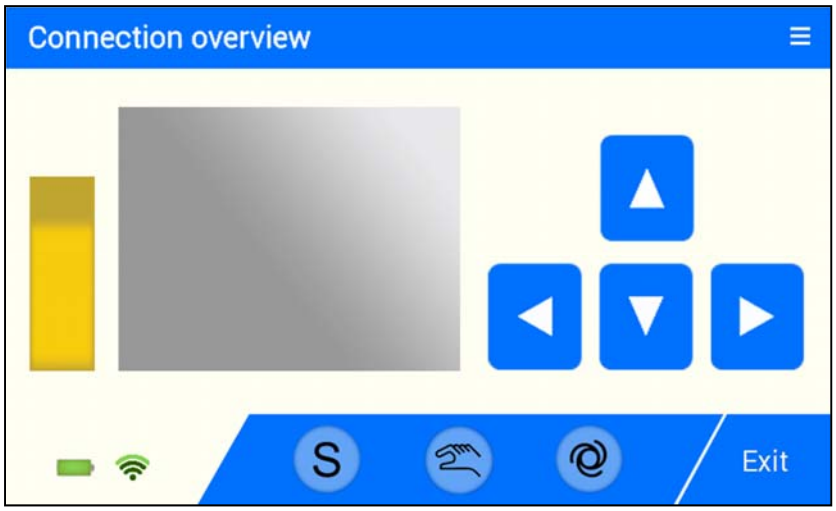
3. Human Machine Interface (HMI)

In order to control our robot as easily as possible we build an android application. This application represents our HMI and provides two different operating modes: the manual and the automatic mode.

In the manual mode it is possible to remote control the robot with the acceleration sensor (provided by phone) and simple joystick buttons. The automatic mode enables the robot to drive autonomously.

With the usage of the library ROS-Java it is possible to publish and subscribe on ROS topics in wireless network.

The following pictures show our HMI. The first picture shows the main menu and second one shows the control menu.



4. Image processing

The detection of the weed plants symbolized by yellow golf balls is solved through a camera based vision system. As a basic approach the obvious features geometry and color are used. Since in a two dimensional image balls are just circles only regions with circle like objects are needed. Therefore the Hough transformation will be applied to the image. After matching the circle like objects against the supposed color range it can result in fragmented sections which should normaly belong together. For a better foreground / background subtraction the Watershed algorithm is used. To eliminate false positives HU's invariant moments are applied to each segmented object. Due to the known dimensions of the golf balls the pose can be calculated through ray tracing.

Frobyte - A Low Cost Platform for Fast Prototyping of Field Robot Applications

Mathias Mikkel Neerup, Anders Kanokpon Garlov Wehland, Michael René Andersen, Anders Nygård Lauridsen

University of Southern Denmark, Maersk McKinney Moller Institute, Odense, Denmark

1. Introduction

Agricultural technology holds a great potential for ensuring competitive and sustainable production, but profound and persistent innovation is required. Therefore as electronics, sensors and advanced software become vital in agricultural production, the need for skilled and specialized engineers and scientists is increased. Introducing engineering students to the challenges within precision agriculture is therefore of great importance.

One subject of particular interest is the field robot, although application at the farm is not imminent. The field robot serves as a very good case for educational purposes, ranging from low-level electronics and mechanics to high-level reasoning and decision making. This unique combination and the robustness required for performing in off-road environments make it both fun and challenging to work with.

At the University of Southern Denmark the field robot has been the subject of many activities ranging from state of the art research projects over design and collaboration courses to student projects. Most activities facilitate the FroboMind software framework (Jensen *et al.* 2012) meaning that the same software is run on the student platforms as on the full scale research robots. This reduces development time, makes prototyping much easier and it makes it possible for students to work on components without necessarily having to understanding the entire system. To facilitate this, a dedicated prototyping and education platform was developed named the Frobit (Larsen *et al.* 2013).

This work presents a new version of the Frobit improved for off-road operation. The size of the platform has been increased and it has been equipped with larger motors and an on-board computer. The software is still based on FroboMind and thus the main concept of students working on the same software as on the full scale robots remains. The new platform has been given the name Frobyte.

2. Mechanics and Sensors

2.1 Hardware

The hardware is based on a wooden plate with two 3D printed wheels mounted directly on the gear axel of the motors. The motors are of the EMG-49 type with planetary gearbox and encoder feedback. The motors are driven by SimpleH H-bridges connected through control logic to an ATmega168 on a RoboCard. The RoboCard is controlled from the on-board PC via a serial connection.

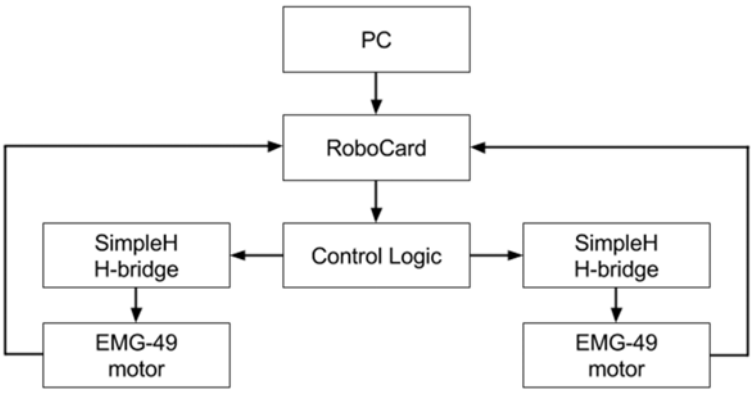


Figure 3 – Electronics

The platform is equipped with a SICK TiM310 Laser Range Scanner (LRS) and a Sparkfun Razor Inertial Measurement Unit (IMU), both connected directly to the on-board PC. All electronics are mounted in a stain-proof plastic box.

2.2 Software

The software is based on Robot Operating System (ROS) and the architecture of FroboMind (Figure 2) thus separating into action, perception and decision making.

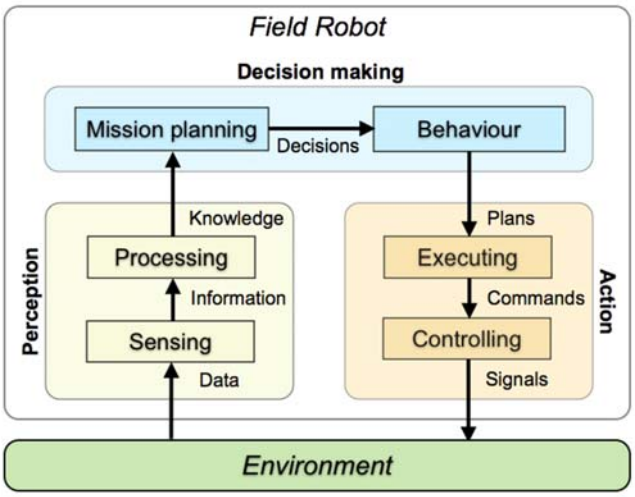


Figure 4 - FroboMind architecture

All software on the Frobyte is running in nodes on ROS. FroboMind is a way to use ROS to avoid one node doing everything or many nodes doing almost nothing. Using nodes, it is possible to make complex functions and still maintain low coupling. When using nodes it is easy to debug and port to other projects.

Perception

The angles and distances obtained from the LRS, are transformed into a point cloud. The point cloud is then split into two point clouds based on the robots heading and each is fed into an instance of the Hough Transform node publishing one row. The information about the two rows is merged together, to calculate an angle error and distance error. Both errors are low-pass filtered to remove noise. The angle error tells how much the Frobyte needs to turn to be in parallel with the rows. The distance error will make sure that the Frobyte stays in the centre between the rows. Combining the two errors makes it easy for the Frobyte to drive smoothly between the two rows.

Action

ROS communicates with the RoboCard through NMEA 0183 encoded messages over UART. The control message contains the desired speed for each of the wheels and the speeds are executed by an optimized version of the Frobit firmware running two PID controllers. The feedback is obtained directly from the encoders on the motors. The wheel speeds are calculated in a ROS node based on desired linear and angular velocities from the decision maker.

Decision making

Figure 3 shows an overview of the state machine for task 1. The state machine is implemented in SMACH which makes it easy to make choices based on the information from the perception layer. Each state tells the Action layer directly what the Frobyte should do at a given time. The Frobyte starts in manual mode, where it can be controlled using a PS3-controller. If the green triangle is pushed, it will transition to

auto mode, which is a sub state machine consisting of several states. When in the follow left wall state, the robot will follow the row until it reaches the end of the row. When that happens, the state machine will transition to the turn state, telling the action layer to turn 180 degrees. The IMU is used to obtain turning information about how far the robot has turned. To stop as accurately as possible it has been implemented as a P-controller.

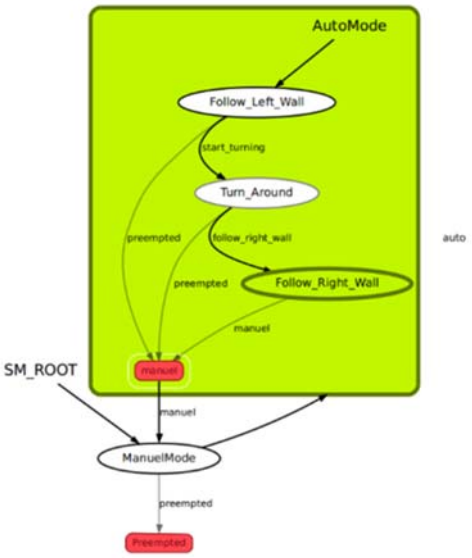


Figure 5 - State machine

3. References

Jensen, K., Larsen, M., & Green, O. (2012). FroboMind, proposing a conceptual architecture for field robots. In *CIGR-Ageng*.

Larsen, L. B., Olsen, K. S., Ahrenkiel, L., & Jensen, K. (2013). Extracurricular Activities Targeted towards Increasing the Number of Engineers Working in the Field of Precision Agriculture . (pp. 1–12).

GardenerRob

Mihaela Tilneac

310507 Arad, Romania

1. Introduction

My interest in robots for agriculture began in the summer of 2006 when I worked in the garden. My mother has made a joke: “Mihaela, build us a robot for removing weeds!”. I decided to bring this fantasy to reality.

2. Mechanics

Chassis from Monstertruck “Detonator” 4WD RtR 2.4GHz.

Drivetrain concept:

- Four wheel drive via cardan shaft;
- Ball-bearing drive;
- Differential in front and rear axles.

Dimension: WxLxH (mm): 340x550x230

Weight: 8 kg

Turning radius (cm): ~30

NR.	IMAGE	DESCRIPTION	QUANTITY	PRICE	USABILITY
1		**Elektro Monstertruck Detonator 4WD RtR** Dimensions (L x W x H): 420 x 350 x 210 mm Weight: ~2 kg (without driving battery)	1	181,37 €	The Monstertruck platform is only used as robot chassis. The electronic components were removed.
2		**DC Motor** **SPEED 600 ECO** Nominal Voltage 7.2 V No-load speed approx. 11000 /min Current 7.5 A	1	15,70 €	This is the main motor. It is used to rotate the driving wheels.
3		**Arduino Uno v3** Microcontroller: ATMEGA328 Operating voltage: 5V Supply voltage: 7V - 12V Digital pins: 14 (6 PWM outputs) Analog pins: 6	4	98,97 €	The entire robot is controlled by Arduino Uno. Four Arduino platforms are used. The first is connected to Pololu Dual VNH5019 Motor Driver. The second is connected to CMUcam4 Shield. The third is connected to Driver L298. The fourth is used to control the robotic arm.
4		**Servo Power HD High Torque 1501MG** Supply voltage: 4.8V - 6V Torque 4.8V: 15.5kg/cm Torque 6 V: 17kg/cm	1	29,09 €	This servomotor is used for vehicle steering.
5		**Servomotor 9g** Voltage: 4.8V - 6.0V Torque 1.4kg/cm	1	8,85 €	This servomotor is used to rotate a distance sensor in front of the robot.
6		**Servo Medium** Supply voltage: 4.8V - 6.0V Torque 4.8V: 2.8kg/cm Torque 6V: 3.2kg/cm	4	50,85 €	These servomotors are used on several applications. The first is used to rotate the distance sensor in rear of the robot. The second is used to rotate the CMUcam4 in order to acquire images from different directions. The third is used to open and close the gripper. The fourth is used to rotate the robotic arm joints.

NR.	IMAGE	DESCRIPTION	QUANTITY	PRICE	USABILITY
7		**Servo Power HD Standard 6001HB** Supply voltage: 4.8V - 6V Torque 4.8V: 5.8kg/cm Torque 6V: 6.7kg/cm	2	32,69 €	This servomotor is used to rotate the robotic arm.
8		**Distance Sensor** **Sharp GP2Y0D810Z0F (10 cm)** Operating voltage: 2.7V - 6.2V	2	13,17 €	These distance sensors are used to detect obstacles on the left side and right side of the vehicle.
9		**Distance Sensor** **Sharp GP2Y0A21YK (10 - 80cm)** Operating voltage: 4.5V - 5.5V	1	13,85 €	This distance sensor is used to detect front obstacle.
10		**Distance Sensor** **Sharp GP2Y0A02YK0F (15 - 150cm)** Operating voltage: 4.5V - 5.5V	2	30,87 €	These distance sensors are used to detect front and rear obstacle respectively. Each of these sensors is mounted on a servomotor which rotates 180 degree in order to collect distance information from different directions.
11		**Pololu Dual VNH5019 Motor Driver Shield for Arduino (ash02a)** Operating voltage: 5.5V - 24V Current: 12A	1	51,98 €	This driver is used to control the vehicle velocity by changing the motor rotation speed.
12		**Driver L298 v2** **Shield for Arduino** Supply voltage: max. 12V Current: 2A	1	17,71 €	This driver is used to control the lamps for optical signaling. Lamps illuminates when weeds are detected.
13		**Cooler for L298N**	1	2,04 €	
14		**CMUcam4 Shield** Open source and re-programmable Arduino Shield Compatible VGA resolution: 640x480 Onboard Image Processing :160x120 Frame rate: 30 fps Operating voltage: 5V	1	97,61 €	This smart camera is used for golf ball detection and weed detection.
15		**microSD Card 4GB**	1	7,47 €	This microSD card is used for CMUcam4 image storage.
16		**Signaling Lamp** Supply voltage: 12V	2	2,27 €	These lamps are used for optical signaling. They illuminate when weeds are detected. The lamps are mounted on left side and right side of the vehicle.
17		**Panic Alarm** Supply voltage: 3V	1	6,58 €	This alarm is used for acoustic signaling. It starts ringing when weeds are detected.
18		**Portable TV 9.8 inch** Supply voltage: 9V	1	68,10 €	This portable TV is used for CMUcam4 image processing monitoring. It is mounted on the top of the robot.

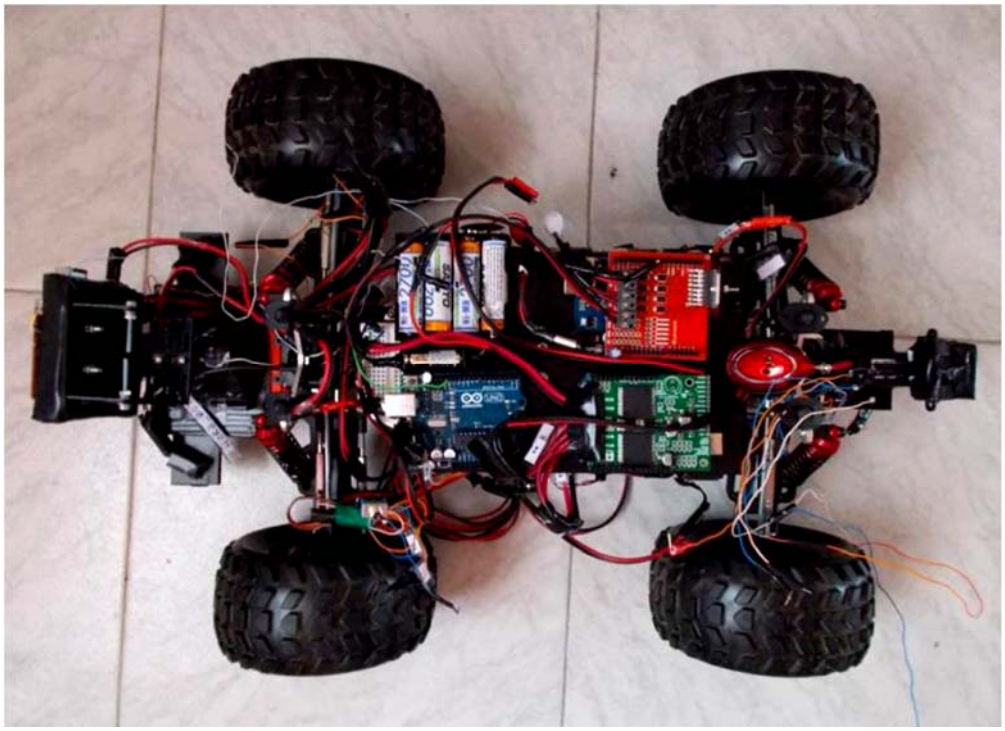
NR.	IMAGE	DESCRIPTION	QUANTITY	PRICE	USABILITY
19		**Pan Tilt**	2	13,17 €	These pan-tilts are used on several applications. The first pan-tilt is used for CMUcam4 rotation. The second is used for robotic arm articulation.
20		**Robotic Claw**	1	13,17 €	This gripper is used for handling objects. (Ex: handling weeds)
21		**Accumulator** NiMH 7.2 V POWER PACK 3000 6 cells	1	22,90 €	It is used to supply the DC Motor SPEED 600 ECO.
22		**Accumulator Charger** for NiMH accumulators 4-8 cells	1	14,90 €	It is used to supply the accumulator POWER PACK 3000.
23		**Battery** 12V, 23AE GP	1	1.4 €	This battery is used to supply the signaling lamps.
24		**Accumulators** 1.2V, Size AA 2300mA/h , SkyR2U 2400mA/h, VARTA 2700mA/h, SANYO	48	141,18 €	These accumulators are used to supply the distance sensors, the servomotors, the CMUcam4, the Arduino boards and the TV monitor.
25		**Accumulator Charger**	1	5,45 €	It is used to supply accumulators size AA.
26		**Adapter Cable**	2	5,90 €	This cable is used to supply Arduino and CMUcam4.
27		**USB Cable A-B**	1	1,57 €	This cable is used to connect Arduino to Computer.
28		**USB to RS232 DB9** **Adapter Cable**	1	7,95 €	This cable is used to connect AgGPSRTK to Computer.
29		**RCA Cable** CMUcam4 - TV connection	1	1,59 €	This cable is used to connect CMUcam4 to the Monitor (TV).
30		**mini BreadBoard**	1	2,27 €	
31		**40 Pin Header Strip** Female	2	2,27 €	
32		**40 Pin Header Strip** Male	2	0,91 €	
33		**Single Wire**	10	1,82 €	

NR.	IMAGE	DESCRIPTION	QUANTITY	PRICE	USABILITY
34		**Cable** 2 wires	8 meter	1,09 €	
35		**Hobbyglas Plate** Transparent	1	1,85 €	
36		**Hobbycolor Plate** Black	1	5,11 €	
37		**Set Screw M3**	1	0,74 €	
38		**Set Nut M3**	1	0,74 €	
39		**Insulation Tape**	3	2,04 €	
40		**Double Adhesive Tape**	5	13,05 €	
41		**Cable Tie**	20	0,91 €	
42		**Rocker Switch**	2	1,18 €	
43		**4 - AA Battery Holder**	12	10,62 €	
44		**Battery Holder Cable**	12	3,54 €	

3. Sensors

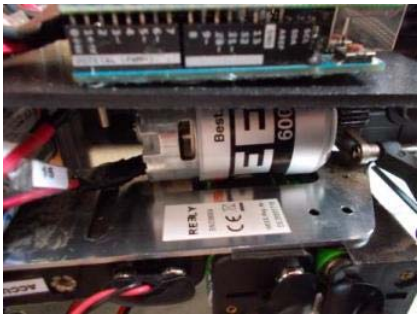
3.1 Hardware

Hardware components and their usability are described in the previous chapter. The following pictures show the components on the robot.

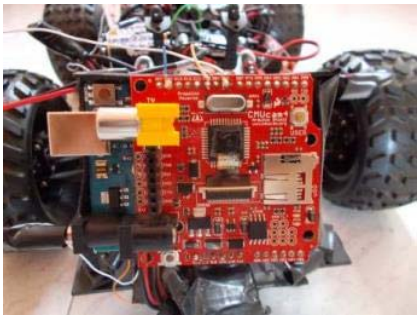



FENDT
GardenerRob


TV & MONITOR
FENDT
GardenerRob








3.2 Software and strategy

Robot software: Arduino 1.0.5.
The Arduino video-tutorials from references [1] and [2] are very useful. They contain some information which helps me on robot programming with Arduino.
Task 1 (Basic navigation):
Navigation through maize rows is based on IR distance sensors. Five IR distance sensors are used. Two distance sensors are mounted on servomotors which rotates the sensors in order to collect distance information from all directions.
Task 2 (Advanced navigation):
The robot must follow an established route through maize rows. Navigation is based on IR distance sensors. Strategy and software are not completed yet.
Task 3 (Professional Application):
A single camera is used to detect weeds (golf balls) on both sides. The CMUcam4 is mounted on a servomotor which rotates left and right. When weeds are detected the acoustic alarm starts ringing and one lamp (left or right) illuminates for a few seconds. The CMUcam4 is a smart camera with image processing and color tracking board.

Freestyle:
The robot must remove weeds by using a robotic arm with a gripper.

4. Conclusion

Turning radius must be reduced. U-turn technique and row counting technique must be improved. The CMUcam4 must be replaced. For indoor applications is CMUcam4 a good image processing board, but it is not suitable for outdoor weed control. The images are strongly affected by sunlight and the camera field of view is narrow.

The robot will be improved in the near future. New pictures and videos will be available on the "GardenerRob" website [3].

5. References

[1] http://www.youtube.com/playlist?list=PLA567CE235D39FA84 [03.08.2014]

[2] http://www.youtube.com/channel/UCZiVMEY5a82Ov2Tq5TvLOhA [03.08.2014]

[3] http://mihaelatilneac1.wix.com/gardenerrob [03.08.2014]

Acknowledgments

Thanks to the sponsor AGCO/FENDT.

I would like to thank several people who have an important contribution in my professional development in the field of agricultural robotics: Dipl.-Ing. Victor Paléologue (École des Mines de Nantes, France), Professor Arno Ruckelshausen (Hochschule Osnabrück, Germany), Senior Lecturer Sanda Grigorescu (Politehnica University Timișoara, Romania) and Professor Valer Dolga (Politehnica University Timișoara, Romania).

HARPER CROP-BOT

Yuyao Song, Xian Liu, Samuel Wane

Harper Adams University, Engineering Department, Newport, United Kingdom

1. Introduction

Currently, there is increasing awareness and development of agricultural robotic system to improve the productivity and enhance the production environment of agriculture around the world. Agricultural robot is also defined as one of the important part of precision agriculture (Emmi *et al*., 2013). Robots have been used for a wide range of agricultural tasks such as weeding, spraying, pruning and monitoring (Frost *et al*., 2000, Smith, 2009, Bakker *et al*., 2010, Kalantari *et al*., 2014). However, the application of robotic system in agricultural farming greatly lags other industries because of complex and volatile work environment and unpredictable natural factors. Therefore, an agricultural robotic system with good expansibility, generality and flexibility should be developed. The objective of this work is to develop an open robotic system for performing different precision farming tasks in a complex environment. This robot is also expected to be integrated in the SAFAR system to promote the development of agricultural robots.

2. Mechanics

The chassis involved in this robotic system is Dagu Wild Thumper 6WD All-Terrain chassis (Figure 1). It features six wicked spiked 120mm diameter wheels, six powerful DC motors and an anodized aluminium chassis made from 2mm thick plate. Each of the six motors are mounted on independent suspension and outfitted with 75:1 steel gearboxes. It could traverse rough terrain and steep inclines and perform tasks in a complex outdoor environment.

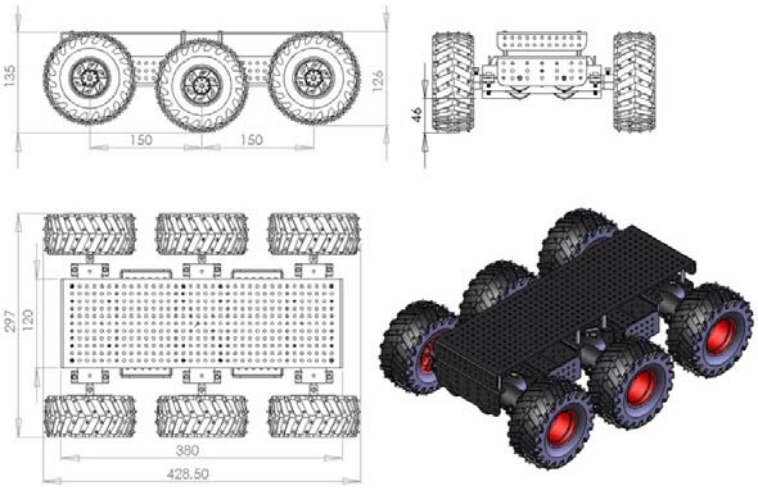


Figure 1: The Dagu chassis of robotic system

3. Sensors

A 2D laser scanner equipped with pulse ranging technology (PRT) and unobstructed 360° vision (R2000, PEPPERL+FUCHS) is used in this robotic system (Figure 2), which is mounted in the central front of the robot. This sensor is capable of very high sample rates and frequencies of 50 Hz making it well suited for a variety of fast applications.

Figure 2: Laser scanner

3.1 Hardware

The computer of present robotic system is fit-PC2i with intel atom Z5xx CPU (1.1 – 1.6 GHz) and up to 2 GB RAM memory (Figure 3). The controller is a dual channel 60A brushed DC motor controller (MDC2260C, RoboteQ), which can be extensively automated and customized using basic language scripts (Figure 4). This motor controller features a high-performance 32-bit microcomputer and quadrature encoder inputs to perform advanced motion control algorithms. The battery equipped on the system is Conrad Energy LiPo Battery (50-4855), while its maximum standby time is 1.5h.

3.2 Software and strategy

Figure 3: fit-PC2i computer

Figure 4: RoboteQ motor controller

For the present robotic system, Microsoft Robotics Development Studio is used as the main software to programme for different tasks. The basis of Microsoft robotics development studio is CCR (Concurrency and Coordination Runtime), which is a net-based concurrent library implementation for managing asynchronous parallel tasks. This technique involves using message-passing and a lightweight services-oriented runtime, DSS (Decentralized Software Services), which allows the orchestration of multiple services to achieve complex behaviours.

For the strategy of navigation in crop rows, laser ranging data points can be identified and a line of best fit which go through relevant data points can be defined as the crop rows on both sides. The robot position can be detected by calculating the distance from the row lines and the angle between robot heading and the centre line. The navigation and direction correction is performed by keeping the robot an equal distance from the left and right sides and within an angle range between robot heading and the centre line, and keeping the robot in the centre line of the rows for forward navigation.

The row end can be detected by the reduction of the number of laser data points on one side and a certain threshold is set to make the robot transfer to the turning mode. A certain rectangular area is defined to calculate the threshold.

The strategy of headland turning is constituted of several phases. The robot firstly performs a U-turn when the row end is detected, when a right U-turn is performed, the end of the right crop line is defined as the reference point. After that the robot drives straight a specific distance and then turns 90 degrees into the next row.

An obstacle is detected as an object which is in front of the robot whilst it is running between the parallel row lines. When there is an obstacle, the number of laser data points reporting a close object is reported, this area is defined by a narrow viewing angle from the robot. A threshold for the number of laser data points and straight distance between the robot and front line can also be set. By comparing the detected data and threshold value, the obstacle can be detected. The robot could then reverse and continue with the following described pattern.

4. Conclusion

This robot is constituted of a unique “super-twist” suspension system, which can ensure a smooth and robust navigation even when driving over uneven or bumpy surfaces. Excellent fittings of 2D laser scanner, portable computer and motor controller contribute to an accurate and reliable operation in agricultural complicated conditions in an automatic mode. The programming carried out with Microsoft Robotics Development

Studio for different tasks involved in the competition makes it possible to further integrate this robot in the present SAFAR system.

5. References

Emmi, L., Paredes-Madrid, L., Ribeiro, A., Pajares, G. and Gonzalez-de-Santos, P. 2013. Fleets of robots for precision agriculture: a simulation environment. *Industrial Robot*, 40 (1), pp41-58.

Bakker, T., van Asselt, K., Bontsema, J., Muller, J. and van Straten, G. 2010. Systematic design of an autonomous platform for robotic weeding. *Journal of Terramechanics*, 47, pp63–73.

Kalantari, D., Shayanmehr, M. and Refigh, A. 2014. Evaluation of the spray generated by a greenhouse spraying robot. *Agricultural Engineering International: CIGR Journal*, 16 (1), pp55-60.

Smith, F. 2009. Let robots do the pruning. *Australian & New Zealand Grapegrower & Winemaker*, 544, pp34-35.

Frost, A.R., Tillett, R.D. and Welch, S.K. 2000. The development and evaluation of image analysis procedures for guiding a livestock monitoring sensor placement robot. *Computers and Electronics in Agriculture*, 28 (3), pp229-242.

HELIOS

Michaela Pußack, Danny Behnecke, Matthias Kemmerling, Hans-Walter Brandt

Technical University of Braunschweig, Institut für mobile Maschinen und Nutzfahrzeuge, Braunschweig, Germany

1. Introduction

The Field Robot Event Design Team joined in 2005 with the objective of developing autonomously developing vehicles. The team consists of students from the fields of mechanical and electrical engineering as well as computer and communications systems engineering.

Our robot Helios has been participating in the Field Robot Event since 2007. As those early Events showed that the mechanical structure was robust and adequate to the tasks, and any emerging soft spots were resolved early on, Helios' mechanics have remained unchanged for several years. The electronic components as well are working reliably and to their biggest part remain as they are described in the 2010 Field Robot Event proceedings [1].

For the last year, the group has focused on developing Helios' software. In the progress of switching to the framework ROS, much of our old code has been revised or replaced. Also, we designed new mechanical equipment for the Robot.

Figure 6: Our robot Helios

2. Mechanics

2.1 The Robot

Helios is a four-wheel driven robot. The high-torque Servomotor (250 W) activates the wheels via one central and two axle differentials. Both axles are steered, allowing for small turning circles. Springs and dampers connect the rigid axles to the body. For further information, please resort to the proceedings of the 2007 Field Robot Event [2].

2.2 Strip-Till Unit

In accordance with this year's contest location, the DLG Feldtage in Bernburg, we wanted our robot to be able to take a bigger part in the plant cultivation progress.

We decided to design a tool for sowing plant seeds. As sowing is highly energy-intensive, the strip-till-method came to our attention as a technique that only requires a part of the field's surface soil to be plowed. We think this is an interesting method for field robots as it enables our relatively small vehicle to sow corn, cabbage, and other kinds of plants that have to be planted in rows.

The attachment we devised is a multifunctional tool. It consists of a rotary harrow for loosening a small strip of earth, a fertilizing unit for adding fertilizer to the turned soil, and a unit for dispensing seeds into the soil. Using a normal plow was not possible because of the robot's small drag force of about 130 N (due to its low weight), so we decided to integrate an active harrow which opens up the ground about 9 cm wide and up to 6 cm deep. The depth is adaptable by adjusting the height at which the module is attached to the robot.

The seeding module contains the seed tank, the mechanism for singularizing the grains, and a blade that places the grain precisely in the middle of the track. It is powered by a servo, so the distance in which grains are placed can be regulated independently from the robot's speed.

The blade is also depth-adjustable for the different kinds of plants.

The fertilizing module places an adjustable rate of fertilizer directly in the grain track.

This technology may not only be interesting for farmers, but also gardeners for sowing in areas without sufficient space for tractors, e.g. greenhouses.

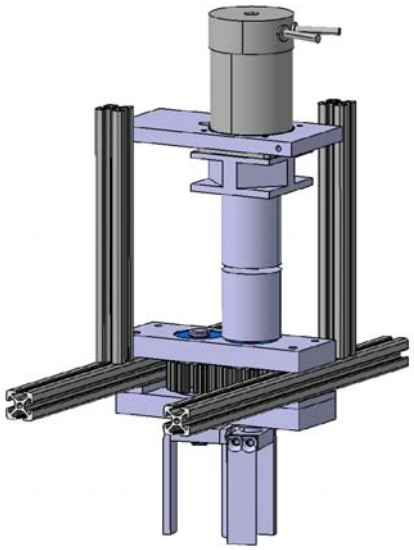


Figure 7: Rotary harrow with motor

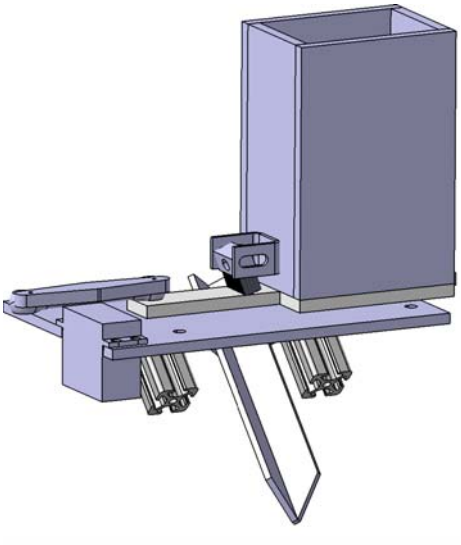


Figure 8: Seeding module

3. Sensors

The laser scanner SICK LMS 100 is our main sensor for navigation. It has a range of 20 m and 270 degrees surround view.

For weed detection, we use two Prosilica GC650C cameras.

3.1 Hardware

The sensor data is processed by an Intel Atom Dual Core mini ITX- board with a CUDA® able GPU3.2.

3.2 Software and strategy

As environment for the synchronization and organization of our system we use the open source framework ROS (Robot Operating System). ROS takes care of many tricky problems like dataflow between parallel running components and gives us due its nature as open source project a wide range of existing programs to use for our project. It works as a middleware between our software and our hardware. Every executable program (called a 'node') is registered in the so called 'roscore' which is the root program that manages all data the nodes provide. These nodes are part of their respective packages. A package in ROS is a compilable division or sub program, having its own make-file and compile-options. Compiling in general is quite comfortable, since the relatively 'catkin' system added an even more tree-like folder structure. Packages are organized by problems they are solving (see below). Commonly these problems are solved in the nodes which are running as threads. These special ROS threads come

with plenty useful extra functions (e.g. frequency regulation) and are all scheduled and managed by the roscore that can be imagined as a sort of data cloud. Nodes can publish on 'topics' in this cloud and subscribe to them to get access to these data. Topics provide information in form of specified messages, a sort of data structures in this context. They vary from simple one-data-type-only to a whole collection of useful data from a node (e.g. it's possible to build a message for image data which contains the relative position, as well as calibration data and some extra information from the image processing like image moments). ROS also provides a system for inter-node-communication, the service concept. It's a simple "I ask - you answer" - every service must have a request and a response specified to define its function. We tried to make best use of these different tools in our implementation and will state how our packets are designed.

Lane Detection

Our lane detection utilizes a RANSAC based approach. With the RANSAC we detect the lanes in our laserscan data and with this information we also estimate the field in which we're driving.

Path Planner

Our paths are basically planned by averaging our found rows to a virtual middle row.

Path Controller

To follow the paths inside the plant rows which we got out of our lane detection, we need to control the steering angle of our vehicle. The controller consists of two parts, a feedforward control and a “virtual drawbar“.

The feedforward control takes the difference of orientation between the robot and the nearest path points and calculates a feedforward control angle. The term “virtual drawbar” means that we take a point with a constant distance in front of the robot and calculate a steering control angle to that point. This process is comparable with the drawbar of a drawbar trailer, where the constant distance is similar to the drawbar length.

The sum of both control values adds up to the steering angle.

If the parameters are well chosen, this controlled system is stable and has a low control error.

End Detection

The ends of the rows are detected with an intersection-check approach. We calculate a vector ahead of our robot which connects the row ends of the estimate field with a line. The odometry vector is much longer than the field and starts in the middle of our vehicle. While we're driving through the rows we check if these two lines intersect. If

they do, we assume that Helios is still in the row; if they do not we conclude that he's passed the row end.

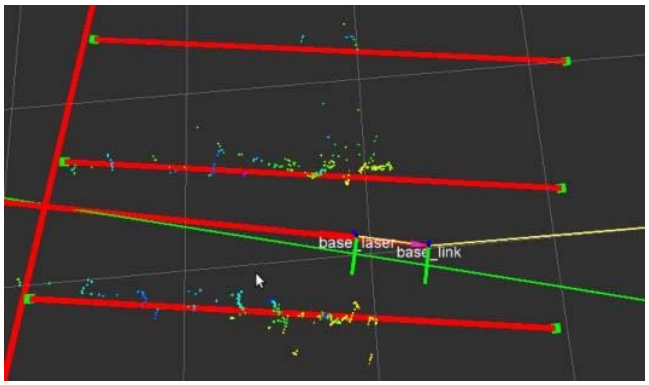


Figure 9: Visualization of laser data, detected rows and row end

Obstacle Detection

With the rows we've detected in the lane detection we construct a polygon (basically a square around our robot) and check the second third of our laserscanner data set (the third that is in front of the vehicle) for points that are inside this polygon, resp. between two rows, and if they are too close to us.

Odometry

At the moment we determine our odometry only by reference to our steering angle, our driving speed and the driven distance. With these parameters and a kinematic model of our robot we calculate our odometry.

Simulation

Testing the robot is usually a time-consuming affair. An adequate environment has to be available, which can be a great problem especially for particular outdoor-applications like driving along field rows. Also, hardware limitations like battery life or possible hardware failures are a regular distraction when testing algorithm performance.

For this reason we decided to use a virtual environment, called Gazebo. The free Gazebo simulator is a 3D multi-robot simulator and was originally designed for the development process of robotic algorithms. It offers a wide range of functionality, like sensor plugins, accurate physics simulation, 3-dimensional graphics rendering and the possibility to design own robots and complex field environments [3]. Gazebo communicates with our ROS system and delivers current sensor data, robot status and simulation time.

We designed a simplified model of our Ackermann-steered robot Helios. Connected to this model are the plugins for laserscanner and cameras as well as the ros_controller package which simulates the steering actuators.

Objects like obstacles, plants or golf balls can be spawned into or removed from the field environment during the simulation.

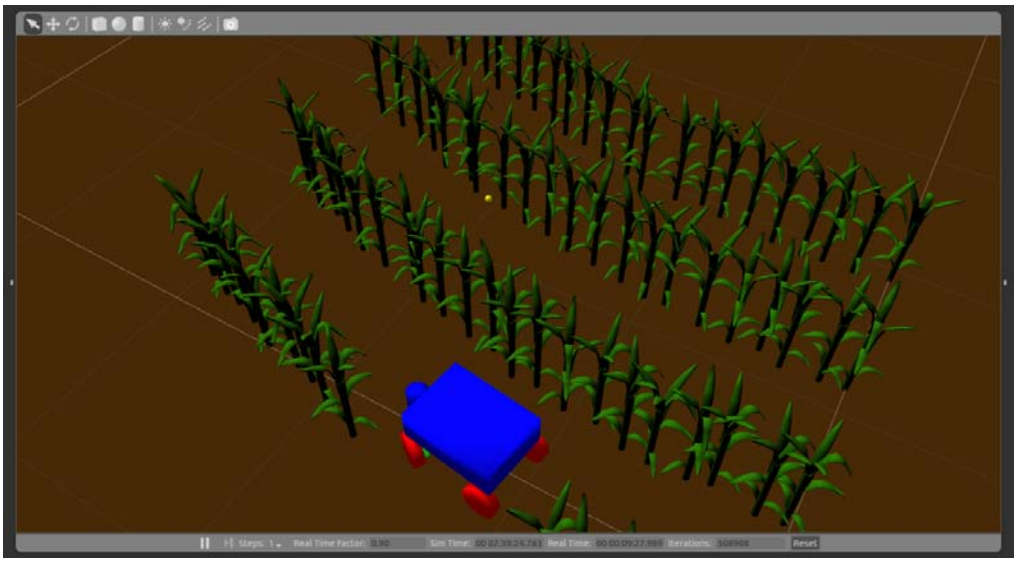


Figure 10: Robot model in gazebo

4. Conclusion

In designing the Strip-Till-Attachment, we discovered that low vehicle weight can be a huge disadvantage. We took great efforts to find a solution that would depend less on the robot's limited powers. The solution we chose brought a variety of new problems along, e.g. the need for a motor and energy supply to power the harrow.

When we started to test the new algorithms, we soon discovered that dividing the code into so many self-sufficient nodes had the disadvantage that to change a parameter, the right node which contains the parameter had to be found first. We then moved on to store our parameters in an extern parameter file. After changing a parameter in this file, the code does not have to be compiled anew, which saved us a lot of time while testing.

Our new RANSAC-based approach to detecting the lines has the advantage that single objects that deviate from the plant row do not distort the lane detection too much. We experienced, however, that the lanes that the robot detects tend to 'jump' around quite a bit, resulting in high steering activity while the robot travels in the field.

In a sufficiently rectangular field, the new end detection works very reliably and with a high success rate of securely entering the new row. In fields where the headland is cut off at a very steep angle, however, the end of the row was sometimes not detected correctly.

5. References

[1]

Proceedings of the 8th Field Robot Event. Braunschweig 2010

<http://www.ecs.hs-osnabrueck.de/uploads/media/Proceedings_8th_FieldRobotEvent_2010_01.pdf>

last accessed 26.05.2014

[2]

Proceedings of the 5th Field Robot Event. Wageningen 2007

<http://www.ecs.hs-osnabrueck.de/uploads/media/Proceedings_FRE2007_01.pdf>

last accessed 27.05.2014

[3]

Gazebo WIKI

<http://gazebosim.org/wiki/Overview>

last accessed 28.05.14

Team Idefix

Frederic Forster, Johannes Hruza, David Lippner, Christoffer Raun, Lukas Locher

Schülerforschungszentrum (SFZ) Bad Saalgau, Überlingen, Germany

1. Introduction

We are four technics enthusiastic students (from 15 up to 17 years), some of us are fieldrobot "newbies" forming a new Idefix-Team and its our second time participating. We meet for some hours on every Friday to do some programming or engineering just for fun. We can use the equipment from the Schülerforschungszentrum. Our teacher is Mr. Locher.

Our robot Idefix has a long history. It was build in 2009 in cooperation with the mechatronics apprentices of ZF Friedrichshafen and pupils of the students research center (SFZ), located in Überlingen. Since then, Idefix took part five times in the field robot event with different team-members of the SFZ.

Some weeks ago, the students of the SFZ-Team in Überlingen got their own class room for their studies.

our new classroom after the preparation for the fieldrobot programming session

2. Mechanics

Two axes from a RC-Model-car are connected to an aluminium frame. Each wheel is mounted with a spring suspension and a shock absorber. Each axis is equipped with a differential gear and driven by a EPOS-controlled Maxon motor with the power of 120W. We selected planetary gears for the motors to reach a maximum speed of 1 meter per second. The wishbones of each axis are connected with some simple levers to two further Maxon-motors for the steering of the front and rear axis. The minimum turning circle diameter is about one meter which is too large to make a U-turn at the end of the maize rows.

The EPOS-controllers of the four Maxon motors are CAN-bus controlled and connected to a linux driven car PC with a USB2CAN-adapter from peak-systems. We use two lead batteries in series connetion to reach a supply voltage of 24 V for the drives and the car PC.

W x L x H (in cm)	50x80x35
Weight (kg)	40
Model/make	Idefix is a Do-it-yourself robot in cooperation with SFZ pupils and ZF trainees
Number of wheels	4
Drivetrain conception	2 Maxon-Motors for front and rear axle
Turning	2 Maxon Motors for front and rear steering
Battery time:	15 minutes

3. Sensors

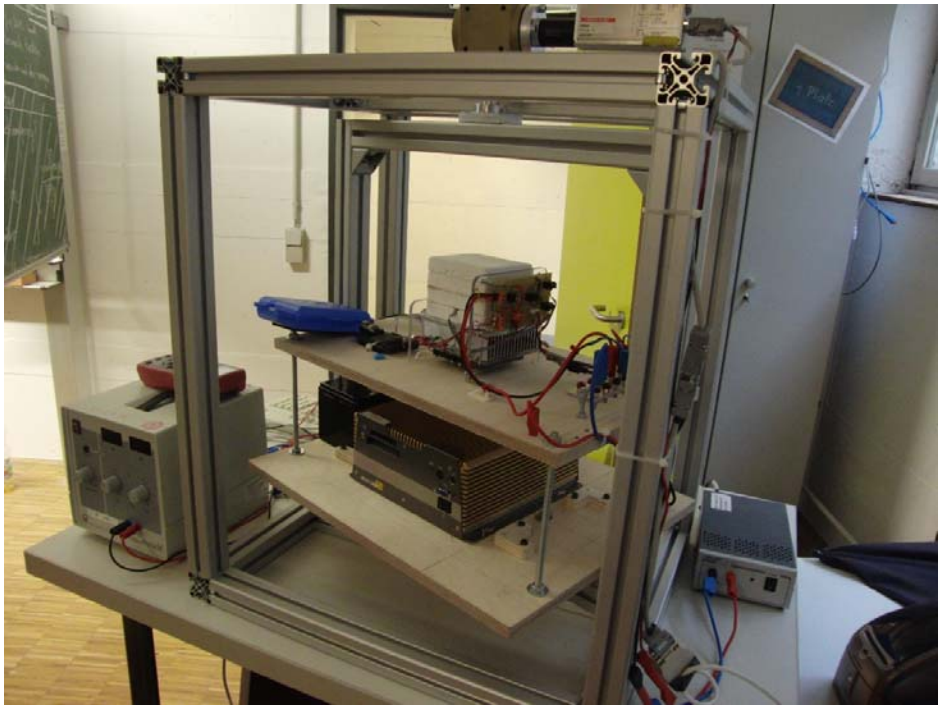
3.1 Hardware

Camera	GigE cam Manta from Allied
Laser	front and rear (Sick LMS100 and Sick Tim3xx)
Gyroscope	Self solderd PCB, based on ITG 3200 chip from Invensense
Odometry	only for driven distance
Teleoperation	Wireless gamepad with USB plugged receiver

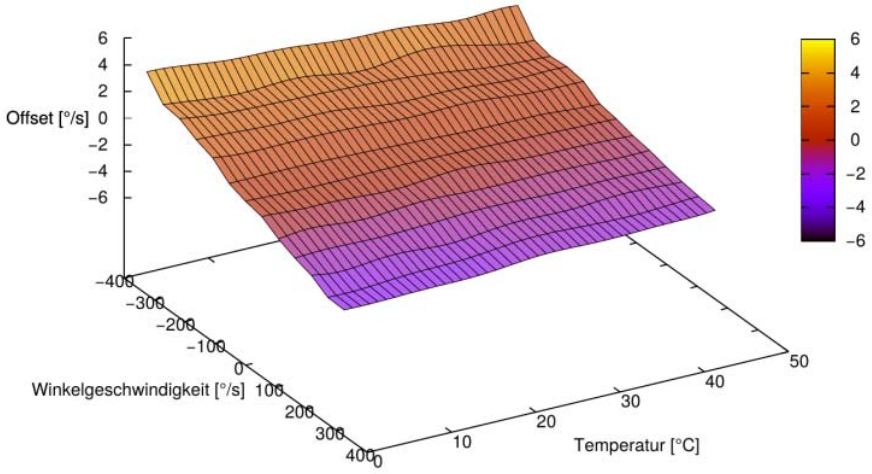
Gyroscope

The ITG 3200 gyro-chip from Invensense is connected via I2C-bus to an atmega 168 microcontroller. The microcontroller sends the data over an UART2USB-Bridge to our car-PC.

To compensate the temperature drift and the influence of the angular velocity of the data from the gyro sensor, we have build an apparatus to cool-down or heat the gyro sensor. We mounted this apparatus in a rotatable frame und did a lot of measurements and data analysing to improve the gyro-data.



Test apparatus for the gyro sensor



Dependence of the gyro offset from temperature and angular velocity

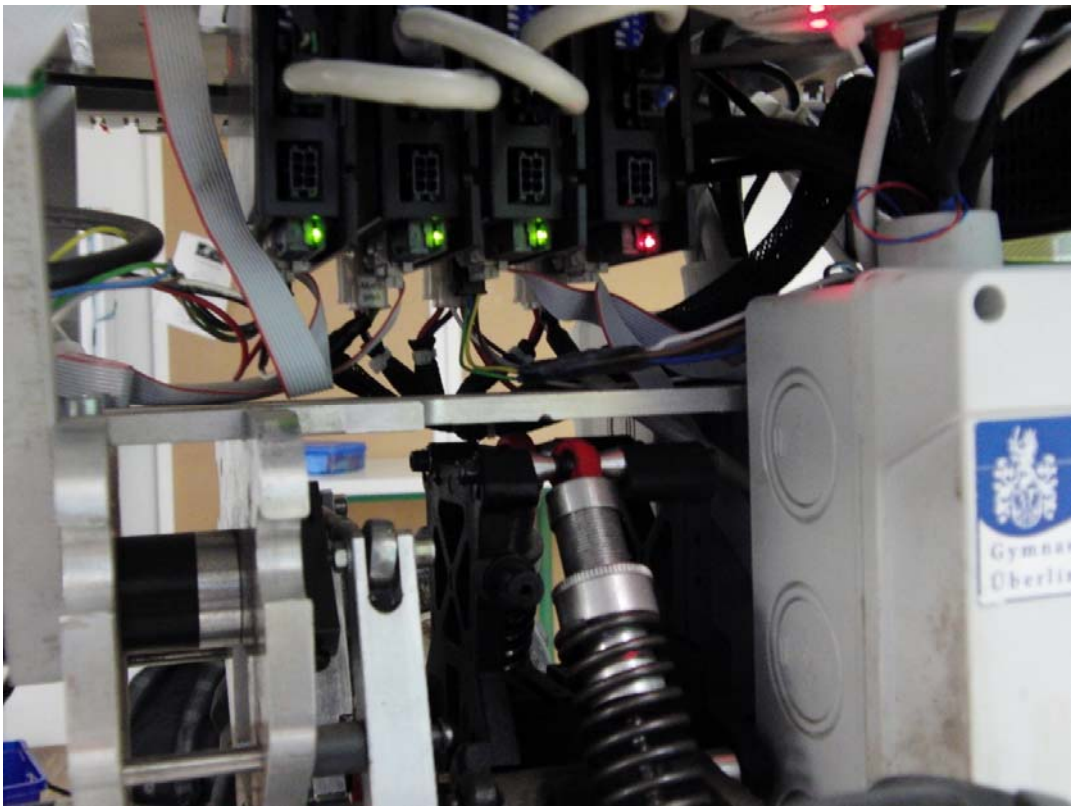
3.2 Car PC

CPU	Dual Core 1.4GHz
RAM	2GB
Operating System	Ubuntu Linux 12.04
Control	Touchscreen, wireless mouse and keyboard

The heart of idefix

3.3 Communication

The car PC is the heart of the robo-control. It communicates over CAN-Bus with the four maxon motor controllers. Sensors are connected to the PC with a USB2Serial-Adapter (Gyro), USB-Bus (Sick Tim 3xx LiDAR), a USB2CAN-Adapter and Ethernet (Sick LMS 100, Camera).



The four Maxon-EPOS motor controllers for the four maxon drives

3.4 Software and strategy

Idefix uses the player robot control framework which is, compared to ROS, not very handy and outdated and a lot of work had to be done to write our own player-drivers for the drives and the TIM3xx laserscanner.

Sick- LiDAR-sensors detect the plants and we create a local 2D map of plant positions. We use a grid for preselection of valid plant positions. Then we feed a proportional controller with a kind of average of the preselected plant positions. This mean-value is used to drive our robot through the rows. That means, if there is a curve and the mean is changing our robot will change its direction. The tricky issue is to find good settings for speed and the proportional constant Kp for the proportional controller. With a LiDAR on the front and rear side, we can drive in both directions. For moving the robot straight ahead in the headlands (task2) we use the (temperature compensated) gyrosensor and odometric data from the drives. For task 3 we tried to do some image processing without real effort.

Because the minimum turning circle diameter is around one meter, which is too large to make a U-turn at the end of the maize rows, we used a Y-shaped move at the end of the maize rows in task 1.

4. Conclusion

After six years of driving, the Maxon motors and controllers proofed as very reliable components. In contrast, the bearings of the wheels, the steering levers and the whishbones are worn out and we believe that this is the last fieldrobot event of idefix. After four years of very fine work, the Sick LMS100 laser-scanner also didn't work as reliably as in his first three years. We believe that the sensor has problems with the bearings. We hope that we can build up a new robot for the FRE2015, feel free to support our new robot!

Kamaro “Betelgeuse”

Simon Merz, Stefan Baur

Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

1. Introduction

Kamaro Engineering e.V. is a student engineering team at the Karlsruhe Institute of Technology. The Team is committed to research and development of autonomous robots. Founded in 2009, the team has participated in all FREs since 2010.

For the past 3 years, the work was mainly focused on the second-generation Kamaro Robot “Betelgeuse”.

2. Mechanics

The Kamaro Robot “Betelgeuse” is a fully custom design. All components are build around a rigid central chassis frame. The drive train formula is 4x4x4, all wheels driven and both axles steerable independently. Both axles feature independent suspension, allowing for maximum off-road-ability. The ground clearance and the stiffness of the dampers can be adjusted.

The mechanical platform is designed to perform well in a wide variety of tasks, allowing the Team to participate in several different events.

3. Sensors

The electronics team is responsible for all electronic components of the Robot. This includes the power supply (external and internal), motors, sensors and some microcontrollers as well as their low-level Software.

Battery management

The Robot is equipped with a self-developed battery management system.

- 2 x 24 V/5Ah Lithium Polymer Akkumulator
- Quick change connectors
- Hot-Plug external power supply
- In-Robot charging
- Single cell voltage monitoring
- Current monitoring
- Temperature monitoring

Sensors

- SICK LIDAR Sensors (front and rear)
- Sonar
- Optical Cameras
- Triple axis magnetometer
- Nine axis motion tracking chip (Gyro, Accelerometer, Compass)
- (Precision GPS)
- multiple mechanical switches
- multiple voltage and current sensors

All Sensors and Motors are connected to a microcontroller (STM) via I^2C, RS232, and CAN. The LIDARs are directly connected to the main computer.

The microcontroller provides an abstraction layer of all the systems on the robot. The main computer communicates directly with the microcontroller.

The Data Processing team takes over at the interface between the low-level microcontroller and the main computer. The main computer runs the computationally intensive algorithms for mapping, navigation and image processing, thus making the actual decisions for the robots movement.

3.1 Hardware

The main computer is a standard x86 machine powered by a PSU for automotive use. The components where chosen as a compromise between power consumption and performance. The main computer is installed in a tight package below and inside the central frame where it is protected from moisture and obstacles. The CPU in use is capable of performing GPGPU accelerated operations (e.g. via OpenCL)

- Zotac Mini ITX Mainboard
- Intel Core i5 Quad Core Processor (Haswell)
- 8 Gbyte RAM
- 32 Gbyte mSATA SSD
- 140 W PSU (9-32 V Input)
- custom air cooling system

3.2 Software and strategy

The main computers software is written in Java now, as it is the most widely used and teached programming language for students of information technology at the

KIT. Substantial performance improvements during the latest Java releases have minimised the gap between C/C++ programs and Java, making it a viable choice for our application.

- Mapping from LIDAR Input
- Sensor data interpretation
- Navigation
- Decision making

THE ROBOT "PARS"

Mustafa TAN[1], Onur Alp SEZER[2], Mustafa DURGUN[2], Muhammed İBİŞ[2], Sefa TARHAN[2], Mehmet Metin ÖZGÜVEN[1], Muzaffer Hakan YARDIM[1]

[1] *Gaziosmanpaşa University, Department of Biosystem Engineering, Tokat, Turkey*
[2] *Gaziosmanpaşa University, Department of Mechatronics Engineering, Tokat, Turkey*

1. Introduction

Since 2003, several teams from the different universities of the world have participated into "Field Robot Event" organized internationally by the prominent universities of Europe every year; and they compete with the robots they developed. We would like to participate into the competition firstly as Gaziosmanpaşa University Biosystem Engineering and Mechatronics Engineering Student Club.

The competition reveals the future vision of precision agriculture. In today's world, the noticeable changes prove that agricultural robots will be used in several fields of agriculture in near future. The agricultural robots run full-automatically in the fields, and this competition proves that will be real. It is the unique agricultural robot competition actualized in open area in the world. The competition also provides opportunities for international cooperation.

The Field Robot Event 2014 competition that will be hosted by Hohenheim University this year will be held in Strenzfeld province of Germany between 17 and 19 June, 2014 together with the DLG Field Days organization.

Our robot called "PARS" was created with a strong team including the graduate and postgraduate students and lecturers from Biosystem and Mechatronics Engineering in order to provide participation into international field robots (Field Robot Event 2014) competition.

2. Mechanics

2.1. Chassis

Our robot will be designed at a mechanic order that will fulfill its purpose as the simplest. In this sense, "Tamiya TXT1" mechanic chassis accepted through its power and durableness will be used (Figure 1,2,3,4). The robot will be built upon this chassis.

Figure 1 Tamiya TXT1 chassis

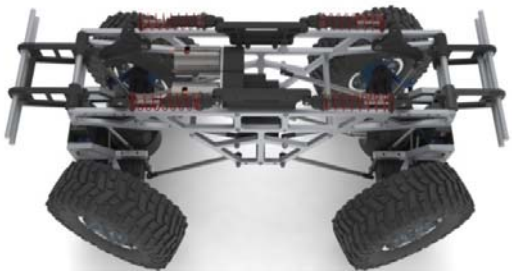


Figure 2 Tamiya TXT1 Chassis top view

Figure 3 chassis CAD drawing

Figure 4 Tamita TXT1 chassis CAD drawing

As the engine, 2 “Servo motors” with 15 kg/cm torque will be used. While choosing this motor, the factors such as the engine speed rate, turnover voltage of the engine, the current drawn by the engine, and size of the engine were considered. In Figure 5 and 6, the engines were presented, and in Figure 7, the power transmission drawings to wheels were shown.

Figure 5 Servo motor

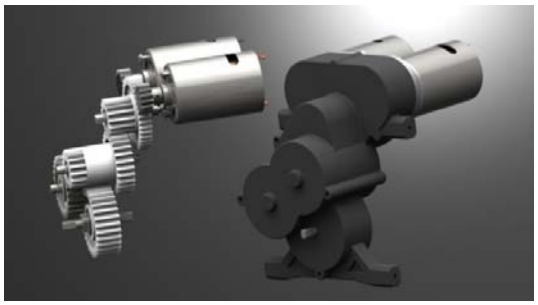


Figure 6 Servo motor power transmission

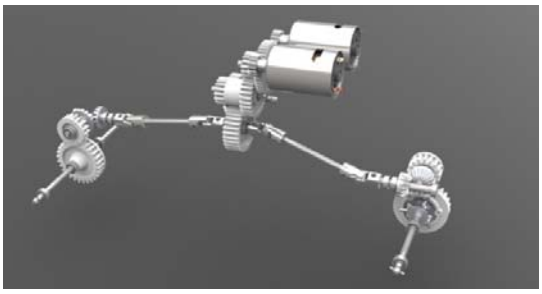


Figure 7 Power transmission from motor to wheels

We will complete the engine casing placing as shown in Figure 8.

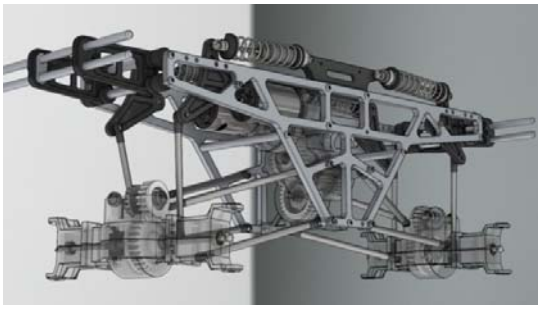


Figure 8 Placement of motors into chassis

2.1 Cover

The robot's exterior is finished with a cover which has a crucial task keeping the electronics dry from the rain. The robot is designed to cover special (Figure 9, 10, 11).

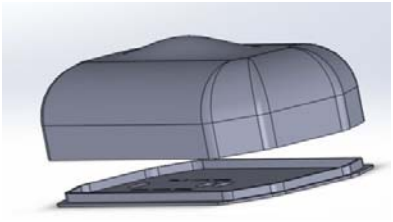


Figure 9 Cover CAD drawing

Figure 10 Cover manufacturing step

Figure 11 Cover manufacturing step

3. Sensors

The robot should have sensors providing it to have peripheral communication in order to make decisions in accordance with its purpose. The robot will include HC_SR04 ultrasonic sensors (Figure 12) as the leading.

Figure 12 HC_SR04 ultrasonic sensor and application

The HC-SR04 ultrasonic sensor uses sonar to determine distance to an object like bats or dolphins do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package. From 2cm to 400 cm or 1” to 13 feet. It operation is not affected by sunlight or black material like Sharp rangefinders are (although acoustically soft materials like cloth can be difficult to detect). It comes complete with ultrasonic transmitter and receiver module.

Features:

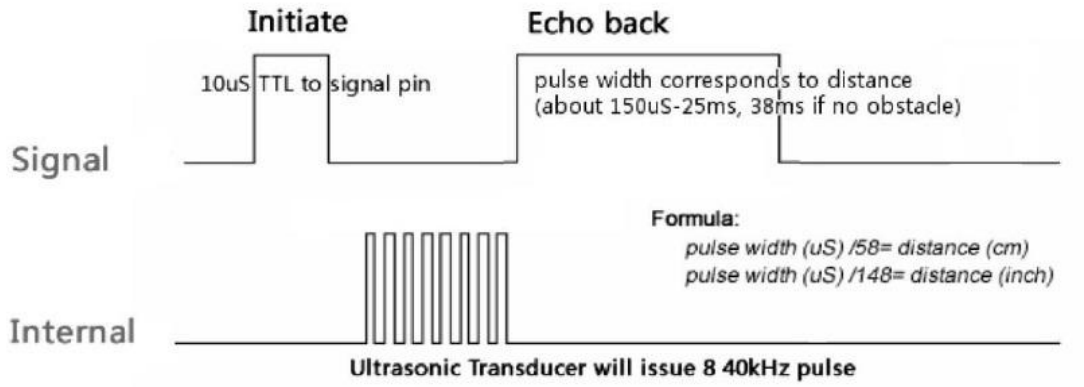
- Power Supply :+5V DC
- Quiescent Current : <2mA
- Working Currnt: 15mA
- Effectual Angle: <15°
- Ranging Distance : 2cm – 400 cm/1" - 13ft
- Resolution : 0.3 cm

- Measuring Angle: 30 degree
- Trigger Input Pulse width: 10uS
- Dimension: 45mm x 20mm x 15mm

The timing diagram of HC-SR04 is shown. To start measurement, Trig of SR04 must receive a pulse of high (5V) for at least 10us, this will initiate the sensor will transmit out 8 cycle of ultrasonic burst at 40kHz and wait for the reflected ultrasonic burst. When the sensor detected ultrasonic from receiver, it will set the Echo pin to high (5V) and delay for a period (width) which proportion to distance. To obtain the distance, measure the width (Ton) of Echo pin [1].

Time = Width of Echo pulse, in uS (micro second)

- Distance in centimeters = Time / 58
- Distance in inches = Time / 148
- Or you can utilize the speed of sound, which is 340m/s



3.1 Hardware

Electronic

"Arduino DUE (Figure 13)" embedded system will be used as the main circuit. And the other peripheral units will be connected to this.

Figure 13 Arduino DUE

The microcontroller mounted on the Arduino Due runs at 3.3V, this means that you can power your sensors and drive your actuators only with 3.3V. Connecting higher voltages, like the 5V commonly used with the other Arduino boards will damage the Due.

The board can take power from the USB connectors or the DC plug. If using the DC connector, supply a voltage between 7V and 12V.

The Arduino Due has an efficient switching voltage regulator, compliant with the USB host specification. If the *Native*USB port is used as host by attaching a USB device to the mirco-A usb connector, the board will provide the power to the device. When the board is used as a usb host, external power from the DC connector is required.

The Due has the ability to change its default analog read and write resolutions (10-bits and 8-bits, respectively). It can support up to 12-bit ADC and PWM resolutions. See the analog write resolution and analog read resolution pages for information.

The Due has expanded functionality on its SPI bus, useful for communicating with multiple devices that speak at different speeds. See the Due extended SPI library usage page for more details [2].

Battery

The main idea in battery selection was easy-changeable battery use at equal sizes. For that reason, we decided to use lithium polymer battery (LIPO) (Figure 14) as 3300 mAh.

Figure 14 Arduino DUE

Cells sold today as polymer batteries are pouch cells. Unlike lithium-ion cylindrical cells, which have a rigid metal case, pouch cells have a flexible, foil-type (polymer laminate) case. In cylindrical cells, the rigid case presses the electrodes and the separator onto each other; whereas in polymer cells this external pressure is not required (nor often used) because the electrode sheets and the separator sheets are laminated onto each other. Since individual pouch cells have no strong metal casing, by themselves they are over 20% lighter than equivalent cylindrical cells.

The voltage of a Li-poly cell varies from about 2.7-3.0 V (discharged) to about 4.20-4.35 V (fully charged), and Li-poly cells have to be protected from overcharge by limiting the applied voltage. The exact voltage ratings are dependent on the materials and manufacturing technologies used and are specified in product data sheets [3].

3.2 Software and strategy

While writing the software, the algorithm we organized, and input and output pins of the microcontroller in our electronic circuit should be considered. We are going to use the Arduino DUE circuit board. We can easily program the Arduino microcontroller with libraries. The code we wrote will be converted into machine language through ARDUİNO compiler, and will be transmitted into microcontroller through the programmer.

The robot will compete at Task1, Task2 and Freestyle.

The main algorithms and methods of the robot will be Line perception, Line break perception and Return skills. The study plan will be as such. Firstly, a simple and easy-to-use method will be developed. Then, a more improved method will be developed providing the input and output interfaces of the algorithms to remain as the same. Doing such, it will be easier to learn what the robot should perform. So, we will analyze the aforementioned algorithms and methods, and choose the method providing the optimum result.

Robot Pars Programming and Algorithm

Robot PARS is using Arduino Due Card for controlling servos and ESC brushed motor. Our first challenge is get work ESC and probing values on it.

```
void arm()
{
  Serial.println("5 icinde basliyor.");
  delay(5000);
  Serial.println("Arm ediliyor.");
  setSpeed(30);
  delay(2000);
  setSpeed(90);
  delay(2000);
  Serial.println("Arm edildi.");
  setSpeed(30);
  delay(2000);
}
```

That part is our arming to ESC motor and next part gonna be adding moving forward and moving backward programs parts.

```
void ileri() {
  servo_on.write(90);
  delay(15);
  servo_arka.write(90);
  delay(15);
  esc_ileri();

  //esc değeri yazılacak//
  // servo deattach durumu yazılabilir //
  }
 void geri() {
  servo_on.write(90);
  delay(15);
  servo_arka.write(90);
  delay(15);
  frenleme();
  esc_geri();
  //esc değeri yazılacak//
  // servo deattach durumu yazılabilir //
  }
```

That part is only 2 seperate program allow to moving forward and moving backward. Also we added another 2 programs like this to doing moving right and left.

```
// Sensor-1//
#define trigPin1 22
#define echoPin1 23
// Sensor-2//
#define trigPin2 25
#define echoPin2 24
// Sensor-3//
#define trigPin3 26
#define echoPin3 27
// Sensor-4//
#define trigPin4 28
#define echoPin4 29
// Sensor-5//
#define trigPin5 30
#define echoPin5 31
// Sensor-6//
#define trigPin6 32
#define echoPin6 33
// Sensor-7//
#define trigPin7 35
#define echoPin7 34
// Sensor-8//
#define trigPin8 37
#define echoPin8 36
//////////////////
```

Our algorithm is very simple. We have 8 sensors and all the sonar sensors has calculation for SR-H04 and finding distance value in “cm”. Next part we added to start button command for when race start we switch robot will move and change that switch will stop the robot. Our next part is movement part of our programming parts. According the constest rules and books we know distance of plants and height and width values according to that we calculate distances also added our robot height and width too because that also effect to our sensors and also give error (-/+) for that. That's provide to extra space and extra accuracy for sensors. Distances was nothing but sensors malfunction problem and be killer.

Front sensors algorithm;

Our front number is 3 and position is cross left,right and center. Cross left and right gonna detect plants and give us signal to for moving forward and make decisions for turning left and right part.

Left and right sensors algorithm;

Left and Right sensors make our robot keep in track and also when they detect too much far distances that's mean we gonna have turn left or right depend on position and program

Backside sensors algorithm;

That sensors was putt for only task 2 because when we have to go back from obstacles we gonna need to us our front and and that sensors will be our front side for that movement and go backward and turn left or right depend on position and task 2 algorithm.

4. Conclusion

The main conclusions of the study should be presented as the outcome of the performance putting emphasis on the advantages and disadvantages of the robotic system.

It is not an easy task to make an autonomous robot to work in an unconstructed environment. Although the area where the robot is supposed to move is known, there are still huge variations in the size and form of maize plants. Also, there might be variation in row width, some plants can be missing and abnormal situations can occur. Weather conditions may change while the robot is moving. All these factors are things that must be taken into account when one is designing a robot to such areas.

Our solution was to make as much alternative algorithms and sensors that were possible and reasonable. Adaptation to current situations was also used. Downside in this approach was huge amount of parameters to be tuned.

The whole robot is quite complex system with all the subsystems and algorithms. The huge number of tunable variables in every stage of the robot is a real challenge for testing. The tuning phase on the field requires a controlled practice as all the parameters cannot be tuned at the same time. Therefore during the development of the algorithms the tuning procedure for the parameters was already considered.

It was known that there is very limited time to make the final tuning during the competition warm-up day. The well planned testing and the identification of parameter relations was one of the key factors to make a well-performing robot, especially with limited testing possibilities. As a result of this competition, the University of Biosystems Engineering and Mechatronics Engineering Students will participate for the first time. In a short time, and have worked with a small team. Building the robot from scratch was seen as an important educational perspective.

Acknowledgments
Thanks to our sponsors:

5. References

[1] Cytron Technologies Sdn. Bhd. Product User's Manual – HC-SR04 Ultrasonic Sensor. Taman Universiti, Johor, Malaysia. May 2013.

[2] http://arduino.cc/en/Guide/ArduinoDue accessed date : 12-05-2014.

[3] http://en.wikipedia.org/wiki/Lithium_polymer_battery accessed date: 15-05-2014

TU Kaiserslautern

F. Altes[1], J. Barthel[1], V. Dänekas[1], S. Feick[1], T. Groche[1], Al. Klein[1], An. Klein[1], A. Jung[1], L. Nagel[1], K. Sabzewari[1], C. Simonis[1], P. Sivasothy[1], P. Batke[2], J. Hinkelmann[2], V. Leonhardt[2], D. Nshimyimana[2]

[1] *University of Kaiserslautern, Institute of Mechatronics in Mechanical and Automotive Engineering, Kaiserslautern, Germany*

[2] *University of Kaiserslautern, Robotic Research Lab, Kaiserslautern, Germany*

1. Introduction

The mechatronic laboratory is a part of the study study in mechanical engineering at University of Kaiserslautern. Within this course the students get the opportunity to develop mechatronic systems and learn rapid control prototyping. The students are supervised by research assistants of the *Institute of Mechatronics in Mechanical and Automotive Engineering* (*MEC*).Annually the institute offers this course. All participants are divided into small groups, working on special tasks. In regular meetings the group's results are presented and discussed at least twice a month.

The University of Kaiserslautern Field Robot Team 2014 consists of nine students. Moreover the team is supported by few students of the *Robotics Research Lab* which is a voluntary student organization. The students from different departments come together to work on topics like image processing, mapping and simulation. In addition to the funds of *MEC* the robot is sponsored by *John Deere* and *dSPACE*.

2. Mechanics

The hardware of the robot is based on *HurraxYuckon* monster truck drive-train and chassis. New parts were designed by using the CAD software *Siemens NX and* were manufactured by the workshop of the MEC chair. These parts include e.g. brackets for sensors and steering servos as well as new wheels. **Figure 1** shows an overview of the main mechanical parts.

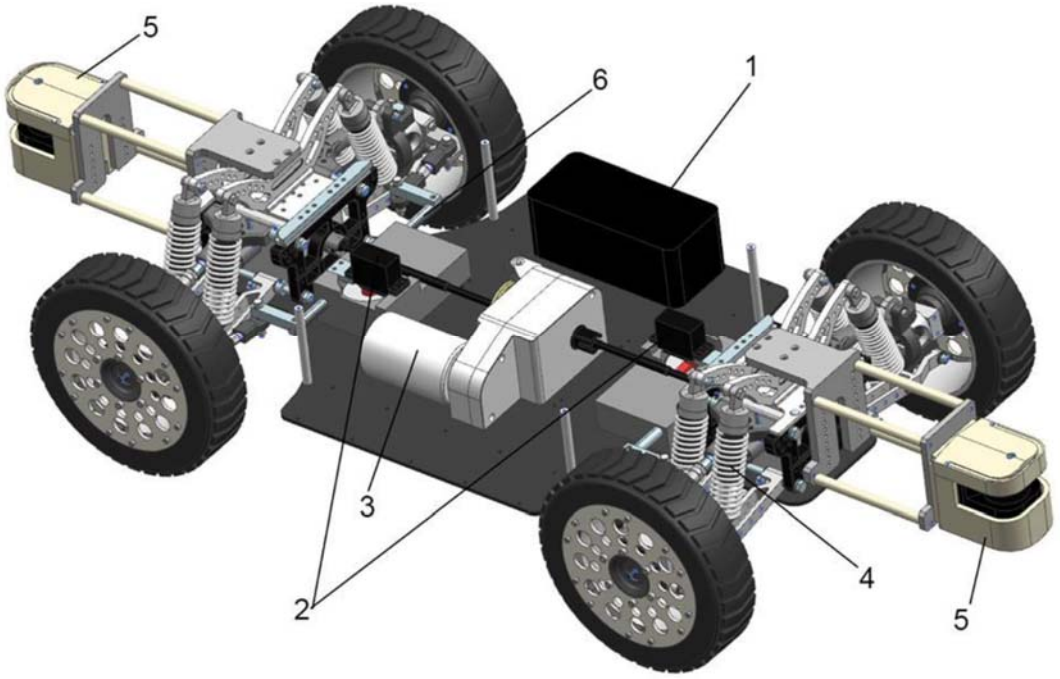


Figure 1 – Overview of the main mechanical robot parts (1) Li-Ion Battery, (2) servos, (3) central brushless DC drive, (4) suspension, (5) laser scanners and (6) steering system

The vehicle's dimensions are 45x112x45 cm and the total mass of approximately 25 kg. The body, which is fully detachable from the robot is made of sheet metal. In Order to reduce the vehicle mass, two carbon fiber plates are used to place the robot components in two levels. The power of the 2 kW *Plattenberg* central brushless DC drive is transmitted to the wheels by a permanent four-wheel drive shaft. Furthermore a *miControl* 100A inverter is used to control the motor in 4-quadrant mode.

The steering system was redesigned by the students to increase the performance which is necessary to navigate the heavy robot on any ground conditions. For this propose *HITEC* servos *HS-M7990TH* are mounted, a gear wheel and steering rack are used to transmit the force to the wheels. Each axle has a separate servo whereby the actuator are operating synchronously. Nevertheless, the minimum turning radius is approximately about 50 cm due to the long wheelbase and limited steering angle. Therefore more than one maneuver are required if the robot has to change the row.

The electric power supply of the robot is carried out by using a lithium ion battery with a total capacity of 8 Ah and a nominal voltage of 33.4 V. Since the electronic parts require different kind of supply voltage, two DC-DC-converters from *TRACO POWER* are used to generate 5V and 12 V for these consumers. Both converters have an approximately output power of 100 W.

A second battery pack allows to swap the batteries and minimizes the interruption during the tests. Figure 2 gives an overview of the electric parts and their different voltage levels.

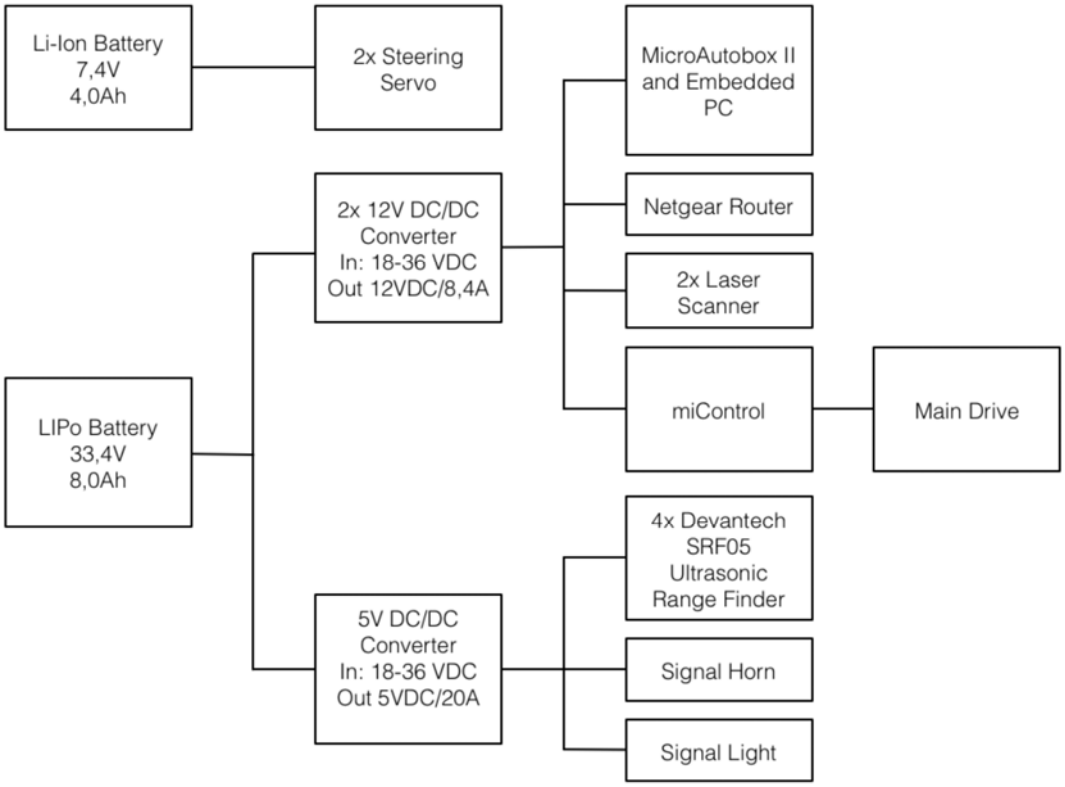


Figure 2 - Overview of the electric supply.

There are also some additional features, such as a piezo siren and a rotating light as well as a 7-segment display. These functionalities can be used to signalize any predefined states e.g. detecting obstacles, or show the distances which are ascertained by the sensors.

3. Sensors

3.1 Hardware

According to obstacles detection, two one dimensional laser scanners are implemented in the front and rear of the vehicle. Each of the two *Hokuyo UBG-04LX-F01* laser scanners deliver a

240° field of view with a maximum range of 5600 mm. Located on the top of the vehicle, four *DevantechSRF05* Ultrasonic Range Finders can deliver additional information about the lateral distances to the environment. The ultrasonic sensors are

not used for the field robot event and may apply for other proposes beyond the competition. Both, the laser scanners and the ultrasonic sensors are mounted adjustable, so that their view can be easily adapted to the current environment.

For some of the maneuvers such as turning, it is necessary to measure some further parameter e.g. yaw rate. Hence, an inertial measurement unit from *Continental* is installed, which is able to measures yaw rate, lateral, longitudinal and vertical acceleration. In order to detect certain objects for the professional task, two standard webcams are mounted. They are located on a frame construction on top of the robot and are orientated to the right- and left side of the robot. In addition to these sensors for autonomous navigation, the vehicle carries a WLAN interface and an R/C receiver for development and testing purposes.

Electronic control unit

For signal processing and robot control, the vehicle uses a *dSPACE* rapid control prototyping hardware. This unit includes the *MicroAutoBox* II (MAB-II) with an embedded PC, which is a compact system with two hardware units. While the actual control functions are being computed on the real-time prototyping unit of the *MAB-II*, additional applications like camera-based object detection and receiving laser scanner data are carried out by the embedded PC. **Table 1** lists the main specifications of the MAB-II of the Embedded PC.

Table 1 - main specification of the electronic control unit

MicroAutoBoxII		Embedded PC	
processor	PPC750GL Power PC	Processor	Intel Atom N270
clock frequency	900 Mhz	clock frequency	1,6 Ghz
global RAM	8 MB	memory solid-state drive	2 GB RAM 64 GB
located RAM	16 MB		
flash memory	16 MB		

The algorithm for row detection, path tracking and headland turns as well as the sequence control of the entire vehicle are essential parts of the software, running on *MAB-II*. The multitude of I/O interfaces makes it easy to connect different kinds of sensors and actuators. To develop the algorithms and data acquisition on *MAB-II,* the software *MATLAB / Simulink* and its toolboxes are used. The software comprises hardware required toolboxes such as *target link* and *embedded coder* as well as software features e.g. fuzzy control. The algorithms are developed in *MATLAB* owns language as embedded functions and converted automatically to c-code using the toolboxes descripted above.

The Image processing and the machine vision are some of the important tasks for autonomous vehicles. The free C/C++ software library *OpenCV* has a comprehensive collection of up-to-date algorithms and is available for various platforms, including Windows® and Linux. The data transfer between the two platforms is carried out by

using the internal Gigabit Ethernet switch and UDP/IP. This solution guarantees a fast communication between *MAB-II* and the Embedded PC which is running on a Linux distribution. Figure 3 shows the structure of the used signals and communication between the different parts of the robot and the central computation unit.

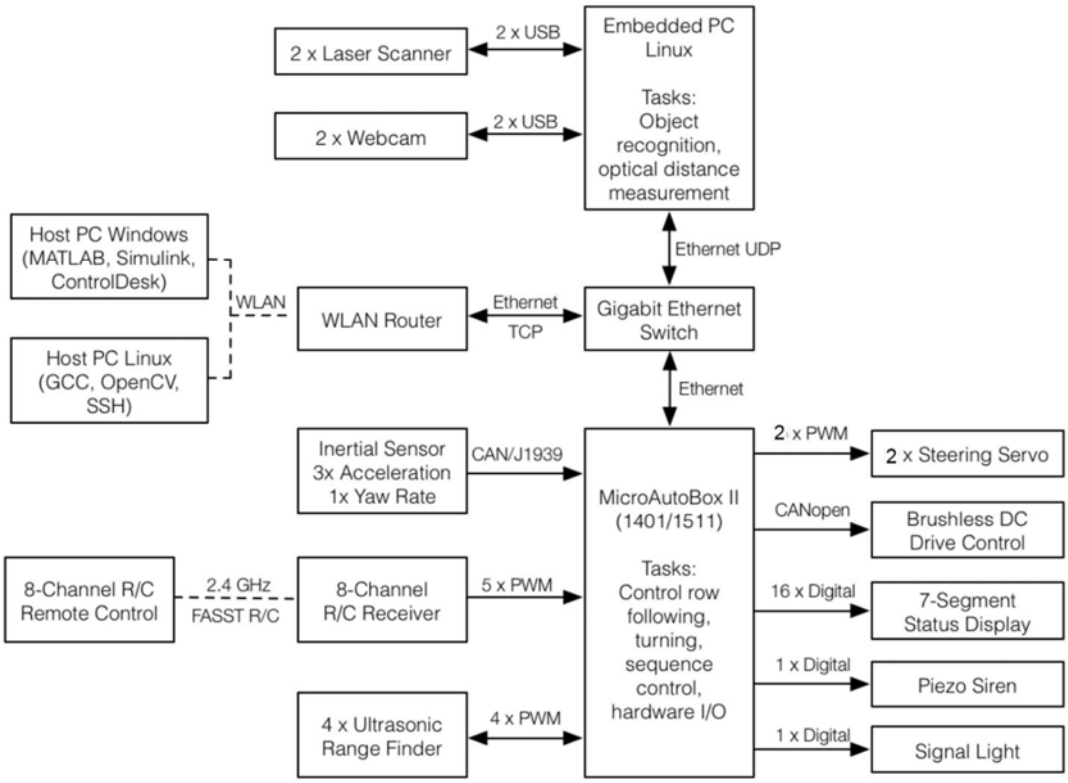


Figure 3 - Signal overview of the robot

3.2 Software and strategy

State machine

Generally the field robot is controlled by a logic for supervisory control and task scheduling, implemented in *Stateflow®* as a state machine. *Stateflow®* is an environment for modeling and simulating combinatorial and sequential decision logic based on state machines and flow charts. The integrated state machine consists of various kinds of states which includes e.g. path tracking, turning maneuver as well as different kind of competition tasks. According to the current task there are also some additional functions, which can be activated. This features are used to detect blocked rows or missing plants by ignoring small gaps between the plants as well as the observation and comparison of the both plant rows (Figure 4). It is also possible to

change the direction of the movement and switch between the laser scanners. Due to the implementation of the State machine, the software can be developed indecently and modular by the students.

Path tracking

Moreover, there are algorithms for path tracking, detection and counting of rows and the turning at the headlands. The distances to the obstacles are measured by the corresponding laser scanner. The laser scanner has a resolution of ca. 0.3° and is able to map the robot environment in detail. The preprocessing scripts are run, and the relevant information are filtered and prepared.

Based on this information a desired position with a predefined distance to the robot is calculated. To improve the algorithm of path tracking and make it more robust against all kinds of field and plant conditions, a fuzzy control system is utilized. The plant rows are counted by detecting the gaps between the plants if the software runs in relevant state.

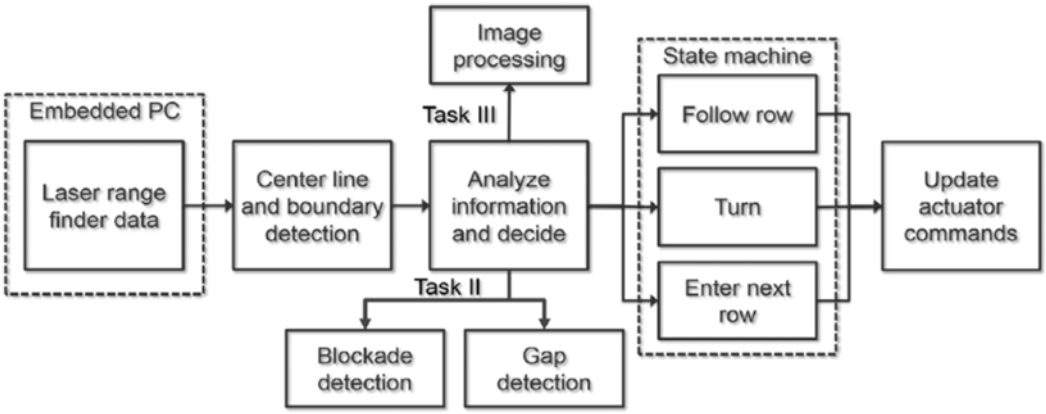


Figure 4 - Software structure of the robot

Image Processing

One particular challenge of the Field Robot Event 2014 is to have the autonomous robot identify a randomly located yellow golf ball. Recognizing the golf balls is performed sequentially by:

- identification of image areas containing a high proportion of yellow color
- measuring the distance between the vehicle and the obstacle by using the depth information of the stereo image and the size of the discovered shapes
- localization of the ball and indicating on the map

The communication between the two platforms is modified to command the robot from the Linux platform, due to the fact that the webcams are supported only by the embedded PC which has USB interface.

4. Conclusion

Within this course the students get the opportunity to learn about team work, self-organization and project management. Participating in the Field Robot Event means an additional motivation for the students to manage their tasks and achieve the goals. Using the described hardware basis allows the students to learn techniques of rapid control prototyping as well as designing new parts and implementing new actuators and sensors.

The robot of University of Kaiserslautern has been developed during the last three years. However there is a lot of optimization potential, which could be exploited in the future. The large turning radius restricts the agility of the robot and could be decreased with a new design. Innovative algorithms such as artificial neural networks and improvement of fuzzy logic are some of the challenges, which could be implemented to increase the quality of the software. The MEC will offer this course in coming years and aims to participate in this competition with new students and better performance.

Team Phaethon – University of Siegen

Klaus Müller, Reza Behrozin, Sven Höhn, Jan-Marco Hütwohl, Thomas Köther, Timo Rothenpieler, Florian Schmidt, Tim Wesener, Whangyi Zhu, Klaus-Dieter Kuhnert

University of Siegen, Institute of Real-Time Learning Systems (EZLS), Siegen, Germany

1. Introduction

The 12th Field robot event in Bernburg, Germany was the second competition for the team of the University of Siegen. After the last year's Field Robot Event, where the Team Phaethon had already won the third place in the professional task, the team of this year succeeded in the first four tasks (three first places, a third place) and became overall winner of the competition.

The current team consists of eight students, of which three students took part a second time. The students are studying computer science and electrical engineering in bachelor and master degrees.

Fig. 1: ***Team Phaethon.*** *(left to right) Klaus Müller, Tim Wesener, Jan-Marco Hütwohl, Florian Schmidt, Whangyi Zhu, Thomas Köther, Sven Höhn Reza Behrozin.*

The current robot is based on the last year's version but was reworked strongly. Besides the computer hardware the team has changed the complete drivetrain including motors, motor controllers and axles. Additionally the software was changed completely.

2. Mechanics

The main problem of the last year's robot was the missing differential. Based on this, the robot was not able to drive narrow curves satisfactorily. Due to that the current robot got new axles, motors and motor controllers.

2.1 Chassis

Phaethon's chassis is based on parts of different radio-controlled rock crawler and monster truck platforms extended with custom-made parts. The mainframe comes from the rock crawler *RC4WD Super Bully*. The original rigid axle has no differential gearbox and was replaced by *Tamiya Clodbuster* axles. These were improved by ball-bearings and custom-made aluminum knuckles for a higher steering angle.

2.2 Motors and controllers

The robot has two powerful *LRP Vector K4 Brushless-Truck* motors with a maximum power of 80 watt each. They are mounted to the axles to lower the robot's center of mass. Additionally the motor comes with three hall-sensors. The sensors are connected to the motor controller for precise power and throttle setting and also to the embedded board for measuring the rotational speed and direction. The motors are waterproof, fully adjustable and replaceable. [motor]

They are controlled by two *LRP iX8 brushless speed-control* units. Besides the forward and reverse driving mode, the controller supports breaking. Since the robot has different controllers for front and rear axle, different power profiles can be applied. The steering-software module runs on the UDOO-board, which steers the controllers with help of pulse-width modulation signals. [motor_ctrl]

2.3 Servos

For the steering control Phaethon has a *HS-7980 TH High Voltage Ultra Torque* servo by *Hitec* on each axle. It is controlled with pulse-width modulation signal. Depending on the input voltage the pulling load differ between 360 Ncm and 440 Ncm with a regulating time from 0,17 sec/60° and 0,21 sec/60°. [servo]

The servos can be steered independently. Based on that the robot is capable of three different steering modes: single Ackerman mode, double Ackermann mode and crab-steering mode.

3. Sensors

The following sensor description is divided into odometry sensors and environmental mapping sensors.

Odometry sensors

Since one of the team's aims was to improve the odometry of the robot, they add an additional optical sensor to measure position changes. The old method, measuring the motor rotations, was also maintained.

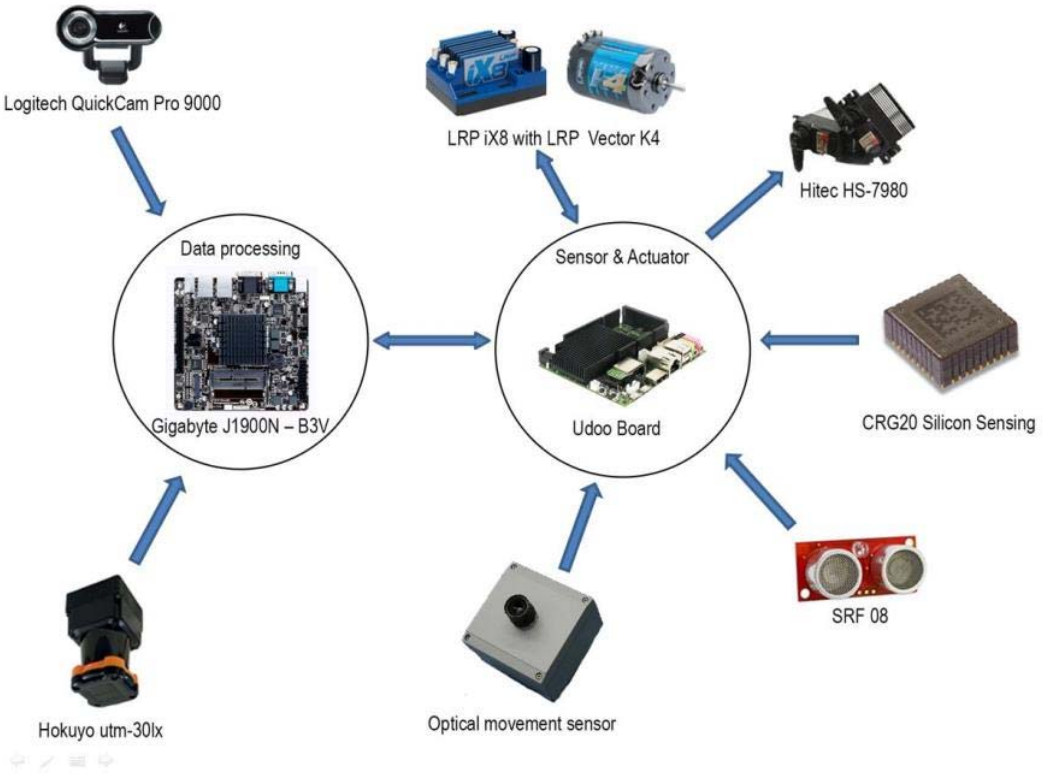


Fig. 2: ***System overview.*** *The computers are connected over ethernet* .

Optical movement sensor

The optical movement sensor uses an optical sensor to analyse the ground and calculate movement based on it. It can measure distances in *x* and *y* direction. To provide a good view even at night, LEDs are mounted to the sensor and light the ground. An integrated ftdi chip makes it possible to communicate with the optical sensor over an USB bus. [obe]

Motor encoders

The LRP brushless motor makes it possible to check the position of the rotor. The hall-sensors of the motors are connected to the UDOO-board, which counts the signals of the three Hall sensors and calculates the rotation direction and speed of the motors and consequently the robot's speed.

Gyroscope

The robot has a *CRG20 Silicon Sensing* gyroscope, which is connected over SPI to the UDOO Board. Its a one axis gyroscope to sense angular movement around the z-axis. [gyro1] [gyro2]

environmental mapping sensors

Laser scanner

The robot is equipped with a Hokuyo utm-30lx laser scanner. This scanner covers an area of 270° with 1024 rays and a frequency of 40 Hertz. The laser scanner was the main sensor, used in every task, for a reliable scan of the environment in front of the robot.

Ultrasonic sensor

To make it possible for the robot, to move backwards, the team used four SRF08 ultrasonic range finder, which are mounted sideways and rear. They communicate via I2C bus.[ultrasonic]

Camera

The used webcam is a QuickCam Pro 9000 by Logitech. This low cost camera is able to record videos with a resolution 800x600 and 30 fps or 960x720 with 15 fps. The maximal resolution is 2 mega pixels.[webcam]

Power supply

Accumulator

The robot is equipped with a package containing four lithium iron phosphate cells with 3.3 V each. They are connected in series and provide a voltage of 13.2 V. As a whole it has a capacity of 15 Ah and makes it possible to drive autonomously for about 3 hours.

The battery voltage is monitored and in case of low voltage the robot emits acoustical warning signals.

Voltage transformer

Due to the fact that the robot needs different voltage levels for its components, a voltage transformer is used to transform the 13.2 V down to 5 V and 12 V. The 5 V circuit is needed to supply the router, the horn and the LED stripes. The 12 V supply is needed to supply the laser scanner, the UDOO Board, the mainboard and the LED of the optical movement sensor.

3.1 Computers

The robot's processing is divided in two different scopes: Sensor readout (UDOO Board) and sensor data processing (Intel Celeron CPU). The robot takes advantage of

both: the UDOO board has a lot of different interfaces and bus-systems; the Celeron CPU is powerful and does all algorithmic calculations.

Embedded computer - UDOO Board

The UDOO Board is an embedded PC equipped with an Freescale i.MX 6 ARM Cortex-A9 quad core with 1GHz and a 1 GB RAM. Next to the Freescale CPU there is an Atmel SAM3X8E ARM Cortex-M3 which can be used with the Arduino software to guarantee an easy access to the GPIO pins. As a whole there are 76 GPIO pins which are used to communicate with the sensors and actuators on a very low level. Above all this there is an Ubuntu operating system, which is connected to the whole system via an Ethernet connection. [udoo]

Mainboard

The second computer hardware is the low power *J1900N – B3V* mainboard with an integrated Intel Celeron 2 GHz quad core processor. It has added 8Gb RAM and a 60 GB solid state drive. Such as on the UDOO Board an Ubuntu operating system is installed. Since it is more powerful than the UDOO-board it is used for software parts which need more computing power like computer vision or point cloud calculations. [gig]

Circuit board

The self made circuit board provides the interface between the GPIO Pins of the UDOO Board and the cable connections of the sensors and actuator. These are:

- horn (controlled over a relay)
- motor drivers and the servo (PWM)
- sonar (i²c)
- gyroscope (spi)
- motor encoders
- emergency stop

Router

As a router/switch we used the *DeLock WLAN Router 11n* which also provides WLAN. This router is mainly used to establish the Ethernet network. The UDOO-board and the mainboard are connected by wire. For programming, configuration and testing external clients can connect via wifi.

3.2 Software

Phaethon takes advantage of the *Robot Operating System (ROS)* framework, which provides an easy, modular way of implementing software and a fault tolerable communication between the different software modules [ros]. Also the inter – module communication isn't bounded to one machine, but can also be done over a network.

So we are able to split the modules, so that the complex algorithms run on a powerful mainboard, while other modules (e.g. the sensor modules) run on an embedded PC. Based on that there are more low level modules which communicate with the sensors and actuators, either with help of libraries for the given interface or directly.

Dead reckoning

The position package implements the dead reckoning. Therefore it uses the gyroscope and the optical movement sensor to calculate the position. This is realised inside a class which can be used in every part of the system.

To calculate the position, the movement detected by the optical movement sensor is rotated around the actual angle of the robot (given through the gyroscope) and finally the result is added to the last calculated position.[reck] So we're able to keep track of the position relatively to a start position(which is usually the beginning of the calculations).

The position package also calculates the speed of the robot with the values given by the optical movement sensor, although this information was only used during development and wasn't used in the final algorithms.

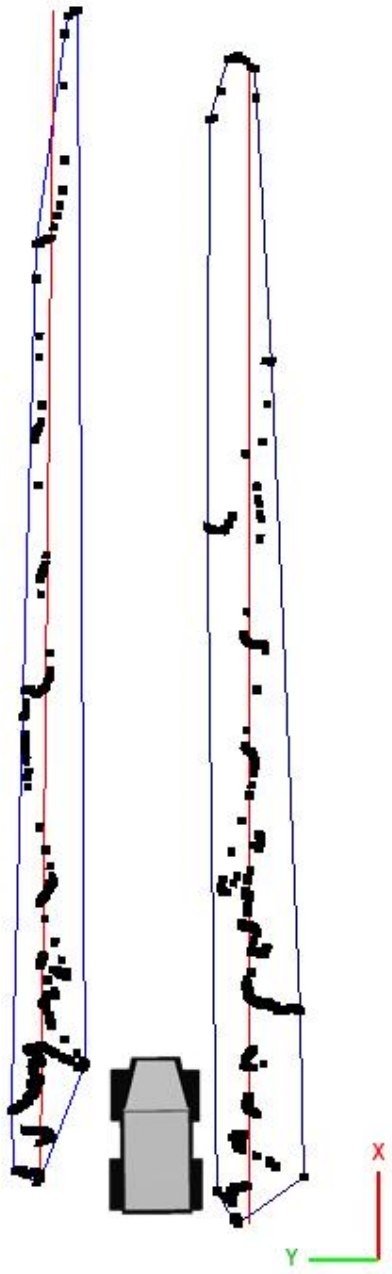
The communication with the sensors is also implemented here. To communicate with the optical movement sensor we used an ftdi library and to communicate with the gyroscope we use the linux spi api.

Navigation in row (task 1)

The navigation_inrow_task1 package is used for navigation within curved rows. The basic elements of navigation consist of a state machine, simple mathematical formulas that determine the steering angle and the last part are termination conditions.

The state machine distinguishes between seven states, three in the rows, three for the change of the rows and one for the stop / reset mode.

In the rows, the sensor data of the laser scanner is used in connection with the scalar product, to calculate the steering angle to the middle of the path. This is possible by assuming that the



distance to one side and the angle are combined to a vector (2D). By doing this for both sides the angle to the center is calculated.

The data of the position package is used while driving through the rows to determine the first part of the row (low speed), the middle part (high speed) and the row's end part (low speed). In the three segments, different steering calculations are used. The row's end is detected with help of the laser scanner. The end detection just runs in the last segment of the row and analyse the rows in the front. If both rows are interrupted for more than one meter, the algorithm detects the row end.

The row changing is mainly based on the odometry information. For finding the next row's beginning the data of the laser scanner is required.

Navigation in row (task 2)

In task 2 a different approach was implemented to keep the robot in the middle of the row compared to task 1. Due to the plant gaps in task 2 the algorithm of task 1 was not sufficient. Hence a clustering method was implemented, to ignore the row's gaps. The clustering is done with the Point Cloud Library (PCL) that comes with ROS. The Point Cloud Library is a framework for 2D/3D image and point cloud processing [pcl].

Fig. 3: ***Scan of task 2.*** *Scan footage of some test plants with the point cloud in black, the convex hull of the clusters in blue and the lines in red.*

Two points belong to the same cluster, if their distance in x-direction is smaller than 1.0 meter and their distance in y-direction is not more than 0.15 m (see Fig. 3). These values have been tuned empirically to get two large clusters – one on the left and one on the right side of the robot, which represent the plant rows. Because of the high threshold in x-direction, missing plants are no problem and the robot is normally able to look up ahead to 5 m in row.

With the help of the found clusters a line is approximated through the middle of each cluster. This is done using linear regression, a simple mathematical procedure for finding the best-fitting line to a given set of points [lin_reg]. The nearest line to the left and to the right is determined and a proportional-integral-derivative (PID) controller tries to keep the robot in the middle of the row.

Another main principle of the algorithm is the dynamic speed control. The robot drives at a fix speed but slows down, if a critical situation occurs. The following critical situations have been defined:

- Only one line was found
- The robot gets too near to one side of a line (Threshold = 10 cm)
- The slope of the lines is too high

The end of the row is detected using the information about the length of the lines. If the length of both lines is smaller than a threshold value, the algorithm stops working.

The position information from the position package is also taken into account, to avoid detecting the end of a row too early.

Changing row algorithm

The navigation_outrow package contains all algorithms for changing rows. This mainly includes a state machine which controls the maneuver, but also a function to find the middle of the destination row and a method for calculating the right steering angle to drive along the rows in a given distance.

The state machine is described shortly in the following:

1. The robot steers out of the row with maximum angle of turn and stops when it has turned 90°.
2. It stores the radius which is driven in the step before. In the following it is assumed that the robot needs the same radius to turn into the next row as well.
3. The distance to the target row's beginning is calculated as following: *(rows_to_skip -1) * row_width - radius*. The robot drives this distance orthogonal to the rows.
4. Since the distance measurement of the odometry becomes more inaccurate for longer distances, the robot searches the final entry of the row with help of the laser scanner.
5. If this succeeds, the robot enters the row. If it does not succeed it goes a short distance forward and tries to detect it a second time. If this succeeds the robot drives a little bit backwards and enters the target row. In case of failure the robot drives in the target row purely by odometry.

In case of changing to the next but one row, the robot is able to do a fast jump over two rows with help of a hard coded steering angle.

Ball detector

The ball detector is used in task 3 and implements image processing steps to find the golf balls. A standard webcam, which is mounted at the left side of the robot, is used for image acquisition and computer vision framework *OpenCV* [opencv] is used for image processing. The ball detection is split in the following steps:

1. image acquisition from camera
2. image transformation from the RGB into the HSV color space
3. thresholding on the HSV image to segment yellow areas
4. segment classification with the help of the known ball size and shape
5. publish the information about the found object

If a ball is found, the current position of the vehicle is read out from an external gps-receiver over RS323 and is written to a file. Additionally the horn and the light are enabled.

For pylon detection in task 2 an advanced version of the ball detector is used. Instead of searching for yellow areas the algorithm searches for white and red areas.

Cooperation Task

Team Phaethon did this task together with the team from Osnabrück and their robot *The great Cornholio*.

The idea was to show communication between two autonomously driving robots. Therefore *the great Cornholio* should act as a robot which stays in a row of maize to do his work. On the other hand Phaethon should be a robot which has to drive through the row to do his work. When the situation appears, that Phaethon has to drive through a row in which the great Cornholio blocks the way, Phaethon should communicate with the great Cornholio and requests to unblock it.

1. Now the following steps happen:
2. Phaethon wants to drive through the row and detects the great Cornholio as an obstacle.
3. Phaethon establishes a connection through wifi and sends a TCP message with the request to unblock the way.
4. The great Cornholio receives this message and clears the way by driving backwardly out of the row, while the Phaethon is following him.
5. After the great Cornholio is driven out of the row and changed into another one, Phaethon is able to change the row, too.
6. Now the maneuver is over and both robots are able to continue doing their work.

The driving in this task was done by the algorithms described above. The algorithm to detect the other robot is based on an existing algorithm, with minor changes had to be done. The TCP connection is established using the program PuTTY.

4. Conclusion

Especially the experience of the last year helped the team to improve the robot and their algorithms. The complete change of the drivetrain greatly improved the driving characteristics and top speed of the robot. The new optical movement sensor provided precise position changes and opened the option to optimize the navigation algorithms accurately. Finally the whole robot system worked well and the team won the first prize.

Acknowledgment

The team Phaethon would like to thank the organizer of the Field Robot Event 2014. Additionally the team thanks their sponsors Ferchau Engineering, Claas Stiftung and Lachmann & Rink GmbH.

5. References

[gig] *GIGABYTE Mainboard GA-J1900N-D3V.* http://www.gigabyte.de/products/product-page.aspx?pid=4918#ov [10.07.14]

[gyro1] *CRG20-00-0300-131_Rev.* http://www.siliconsensing.com/media/287998/CRG20-00-0300-131_Rev_1.pdf [12.07.14]

[gyro2] *CRG20 Digital Angular Rate Sensor.* http://www.siliconsensing.com/media/30649/DocNo-CRG20-00-0100-110-Rev-9PDF.pdf [09.07.14]

[lin_reg] *Linear Regression*. http://www.stat.yale.edu/Courses/1997-98/101/linreg.htm [10.07.14]

[motor] *Vector K4 Brushless Motor – Truck.* https://www.lrp.cc/en/products/electric-motors/brushless/sport/produkt/vector-k4-brushless-motor-truck/details/ [12.07.14]

[motor_ctrl] *iX8 Brushless Speed-Control.* *https://www.lrp.cc/en/products/electronic-speed-controls/competition/brushless/produkt/ix8-brushless-regler/details/* [12.07.14]

[obe] *index | Institut für Echtzeit Lernsysteme: Produkte*. http://www.eti.uni-siegen.de/ezls/produkte/?lang=de [02.07.14]

[opencv] *OpenCV*. http://opencv.org/[12.07.14]

[pcl] *What is PCL?* http://pointclouds.org/about/ [10.07.14]

[reck] Sekimori, D. , Myazaki , F.: *Precise dead-reckoning for mobile robots using multiple optical mouse sensors.* In: *Informatics in Control, Automation and Robotics II* (2007), Springer

[ros] *ROS.org | Powering the world's robots.* https://www.ros.org [12.07.14]

[servo] *Servo HS-7980TH.* http://www.hitecrc.de/store/product.php?productid=21449&cat=310&page=1 [12.07.14]

[udoo] *UDOO: Android Linux Arduino in a tiny single-board computer.* http://shop.udoo.org/eu/product/udoo-quad.html [12.07.14]

[ultrasonic] *SRF08 Ultra sonic range finder.* http://www.robot-electronics.co.uk/htm/srf08tech.shtml [12.07.14]

[webcam] *Technische Daten der QuickCam Pro 9000.* http://logitech-de-emea.custhelp.com/app/answers/detail/a_id/29906/~/technische-daten-der-quickcam-pro-9000 [12.07.14]

TALOS

Florian Balbach, Jan Berlemann, Jannick Coenen, Michael Glaser, Tomás Islas, Christian Jenth, David Reiser, Thomas Wilmsmann, Hans W. Griepentrog

University of Hohenheim, Instrumentation & Test Engineering, Stuttgart, Germany

1. Introduction

In 2014 the University of Hohenheim especially the institute of agricultural engineering will participate on the Field Robot Event with the robot named TALOS. The TALOS took already part of previous events but was redesigned for this year. The main reasons for the redesign were problems with the mechanical driveline and the chassis which were solved with a different amount of motors and a changed frame. The TALOS team consists out of eight agricultural engineering students and was formed in the beginning of 2014. Since then all team members are working together on the project to achieve our goals of a functioning and suitable mobile platform which succeeds the tasks with a satisfactory result.

2. Mechanics

The TALOS is a four wheel drive robot where all components are carried and protected by a rectangular chassis. Different sensors can be mounted to profiled rails which are attached around the chassis and on top of the upper cover. The cover is made of plexiglass to have a direct look onto the inside devices and water resist. In case of assembling the cover can be removed quickly.

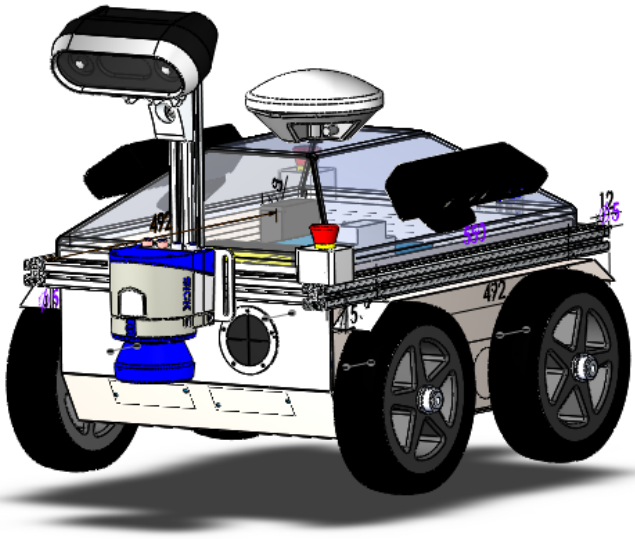


Figure 1: SolidWorks drawing of the robot TALOS with all main elements

The chassis provides enough space for different devices. As the basic frame of the platform it is made of 0.8 mm thick sheet metal. To reach a higher stability the inside is equipped with an additional 2 mm thick frame. In the front and back side has 2 cables channels holes through the chassis and switches openings as well and in the sides has the driveline openings.

A horizontal fixed mounting plate in the inside divides the robot into two levels. The bottom level contains the motors with the encoders and the batteries. On the upper level the electronic devices are mounted. The plexiglass plate is cushioned on four rubber elements for less vibration of the devices.

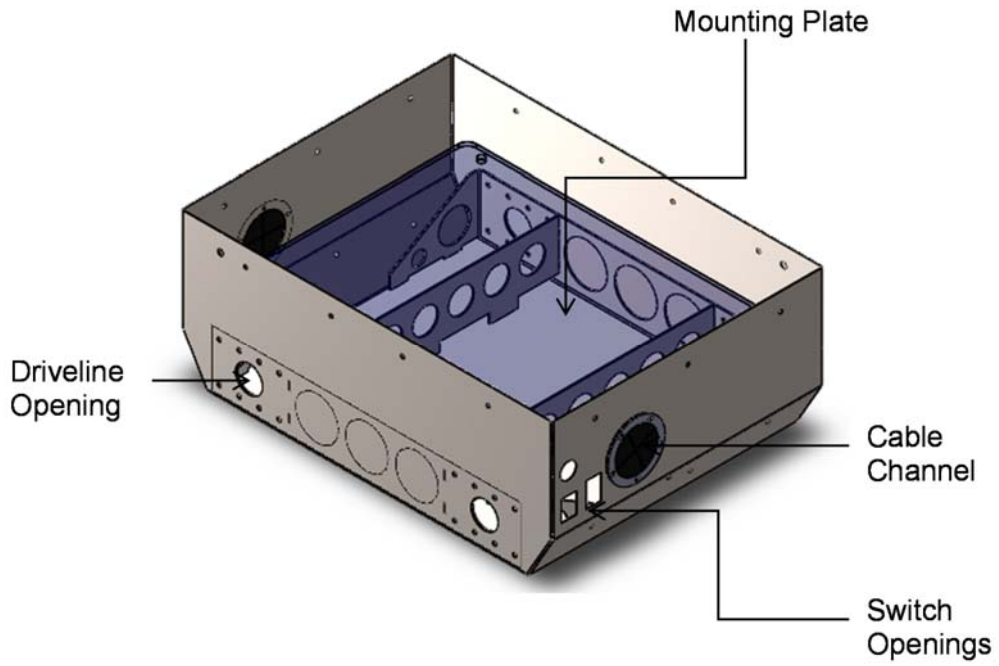


Figure 2: Chassis with openings, inside reinforcements and mounting plate.

The four motors which are used come from Bühler Motors. Those are gear motors of the type 1.61.050.411 and are combined with encoders from PWB. The MEC22 encoders are hollow shaft encoders and provide two square wave outputs for counting and direction information. Our used encoder resolution is 500 counts per revolution. The following table shows the central technical data of the motor [1].

Table 1: Motors technical data [1].

Rated voltage (V)	12
Rated current (A)	3.500
Rated torque (Ncm)	80
Rated speed (1/min)	240
Gear ratio	12
Stages	2
Weight per Motor (g)	1150

The drivetrain is divided in four elements where each motor is directly connected over a bearing house to one wheel. Always two motors are working as one unit which allows the robot to act with a turning radius close to zero.

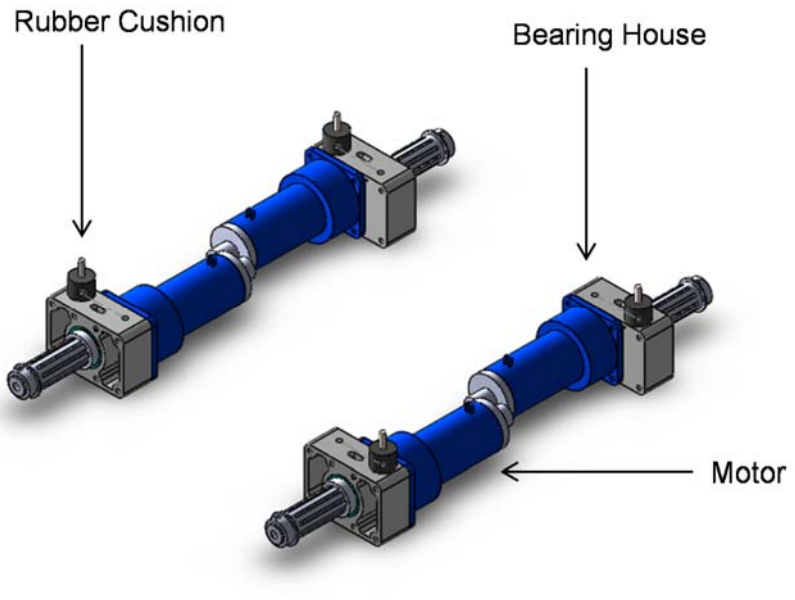


Figure 3: Drivetrain of the TALOS with motors, bearing houses and mounting plate cushions

The bearing house was developed to concentrate the forces on the bearings, thereby reducing the load on the motor shaft; it consists in the main bearing house which contains 2 bearings inside, and also supports the wheel shaft which is divided in 2

pieces, then to transport proposes the outside axle can work without transmit movement to the inside axle that is connected to the motor shaft, this axles will work as unity putting in a pin through both.

The motor have to be screwed with the house bearing, therefore, the motor is fixed in the correct position.

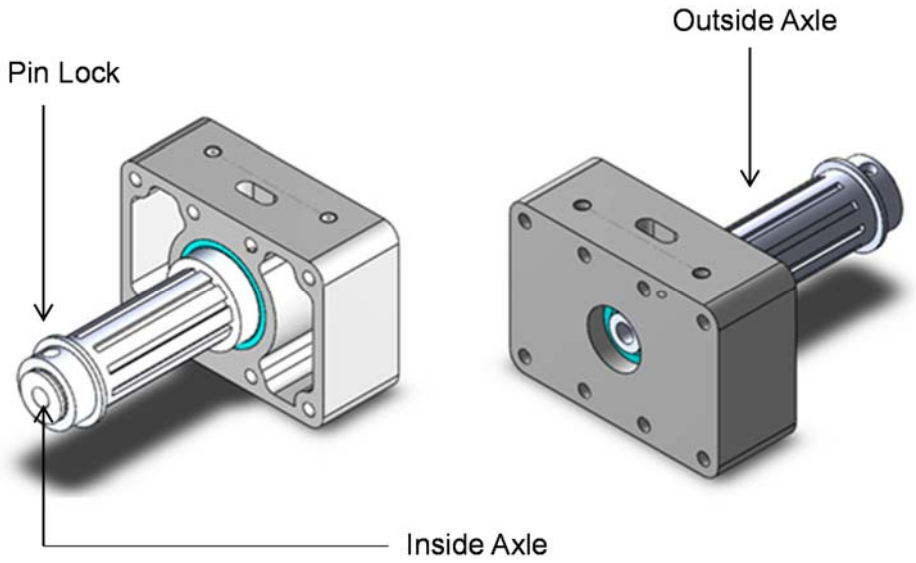


Figure 4: Bearing house with outside and inside axles and pin lock

The robot runs on four full rubber wheels with a diameter of 220 mm. They are selected with a little tread to obviate adherence of dirt and deviance of the Odometry. Tracks are not used due to higher vibrations and needed power demand. The rims are out of duroplast and contain a toothed ring to transmit the forces.

3. Sensors

The TALOS uses the non-contact laser scanner sensor LMS111 from SICK. It is especially designed for outdoor anti-collision environments. The laser has an advanced filtering technology to eliminate false trips in measurement applications. Furthermore it can be adjustable mounted, also able to stand up to wet weather conditions. This sensor offers flexible software that is well suited for a variety of applications, including outdoor environments, robotics and mobile vehicles. The scanning angle of the LMS111 amounts 270° and has a wide variety of interfaces. It supports several applications like collision prevention, navigation support on free-driving vehicles in robotics and path management on automated agricultural vehicles. The output data of the LMS 111 can be easily used for RANSAC-algorithm and further programming [2].

Figure 5: Non-contact laser scanner LMS 111 from SICK.

3.1 Hardware

For data handling TALOS is using an Intel Core i3-2120 processor with a 3M Cache and a clock speed of 3.3 GHz. It uses 2 Cores with an Intel Smart Cache of 3 MB and an instruction set of 64-bit with a maximum TPD of 65W. The graphic processor is an Intel HD Graphics 2000 with a base frequency of 850 MHz and a maximum dynamic frequency of 1.1 GHz [3].

The motherboard is a MSI Z77IA-E53 that supports Dual-Channel mode. IT uses the Intel Z77 chipset with a dual memory channel and 2 DIMM slots. In this the SATA II controller are integrated. Furthermore it contains 2 USB 3.0 ports, 4 USB 2.0 ports, 3 audio ports, 1 eSATA port, 1 VGA port and 1 HDMI port. LAN, WiFi and Bluetooth connections are also possible. The motherboard BIOS provides “Plug and Play” BIOS which detects the peripheral devices and expansion cards of the board automatically [4].

The power supply consist in four sealed lead batteries with 12V/12Ah, connected in parallel, with 2 hours estimated battery time.

3.2 Software and strategy

The TALOS uses the MobotWare which is a standard mobile robot control framework. It includes three core modules. The Robot Hardware Demon (RHD) supports a flexible hardware abstraction layer for real-time critical sensors. The second core module is

the Mobile Robot Controller (MRC) which is a real-time closed-loop controller of robot motion and mission execution. The third core module is the Automation Robot Servers (AURS) which is a framework for processing of sensors and non-real-time mission planning and management [5].

The RHD is based on XML configuration files, which contains the setup parameters for the components and plug-ins to accommodate supported hardware configuration. Besides a network variable database, RHD is also the main real-time scheduler for low level robot control applications, such as MRC. The TALOS uses plug-ins like Stage Simulator 2.1.1, RTK and NMEA GPS interface, Sick laser scanner and the Kinect. The MRC is providing low level real-time control of the mobile robot platform. It uses RHD as an interface to the actual robot hardware. MRC regulates the Odometry, motion controller, SMR-CL interpreter, XML-based socket interface to sensor servers, socket interface to RHD and line and distance sensors. The language for controlling is the Small Mobile Robot Control Language, SMR-CL. It provides the bridge between the hard real-time demands of the control layer and the soft real-time demands of the planning layer. The AURS configuration of the TALOS consists of a camera server, a laser server and a mission management server [5].

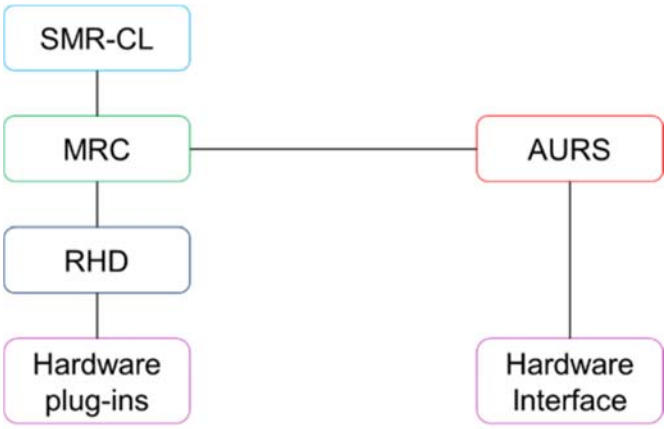


Figure 6: Overview oft the mobile robot control framework (MobotWare) [5]

The navigation strategy is analyze the output data from the laser scanner, defining a “not detection area”, as we know the distance between the rows, therefore, an rectangular area is defined between the rows and 50 cm in front of the robot, when there is a detection inside of the not detection area, then the robot turn a few degrees to the opposite side of the detection. In the end of the row the turn consist in drive forward 75 cm turn 90 degrees to the left or right drive 75 cm forward and turn 90 drees again to the same side like the turn before, after that the actual position is in the next row.

4. Conclusion

Due to its design the robot is very adjustable for different tasks so upcoming generations of students can adopt their needs to the robot. This feature is combined with a compact and reliable construction which makes the robot also interesting for other manifold applications. Even with a relatively high weight of around 30 kg the robot will be able to run under tough conditions because of its four wheel drive. The battery time with around two hours is enough for current tasks but could be improved and upgraded with a different battery technology. The platform provides still room to mount other devices and could be even scaled up in the height. With the moderately simple structure of the Hardware and Software the platform enables to conform to different requirements.

5. References

[1] Bühler, „Bühler Motors,“ May 2014. [Online]. Available: www.buehlermotor.de.

[2] SICK, „SICK Laserscanner,“ May 2014. [Online]. Available: www.sick.de.

[3] Intel, „Intel - Prozessoren,“ May 2014. [Online]. Available: www.intel.com.

[4] MSI, „MSI - Motherboard,“ May 2014. [Online]. Available: www.msi.com.

[5] Beck, *et al.*, „MobotWare - A Plug-in based framework for mobile robots,“ International Federation of Automatic Control, Denmark, 2010.

The Great Cornholio

Kevin Bilges, Fabian Ellermann, Carlo Feldmann, Fabian Lankenau, Andreas Linz, Alejandro Lorca-Mouliaa, Michael Marino, Adrian Merrath, Arno Ruckelshausen, Andreas Trabhardt, Heiko Wilms.

University of Applied Science Osnabrück, Engineering and Computer Science, Osnabrück, Germany

1. Introduction

The following work is intended to describe the hardware and software used by students of the University of Applied Sciences Osnabrück for the 12th annual Field Robot Event. The paper begins with a general mechanical overview of the referred to entry, “The Great Cornholio”, followed by a more in-depth description of hardware used, including specifications, and an overview of the software algorithms used to accomplish general functionality and the applicable tasks for the competition. A conclusion then follows to summarize the findings of the design process.

2. Mechanics

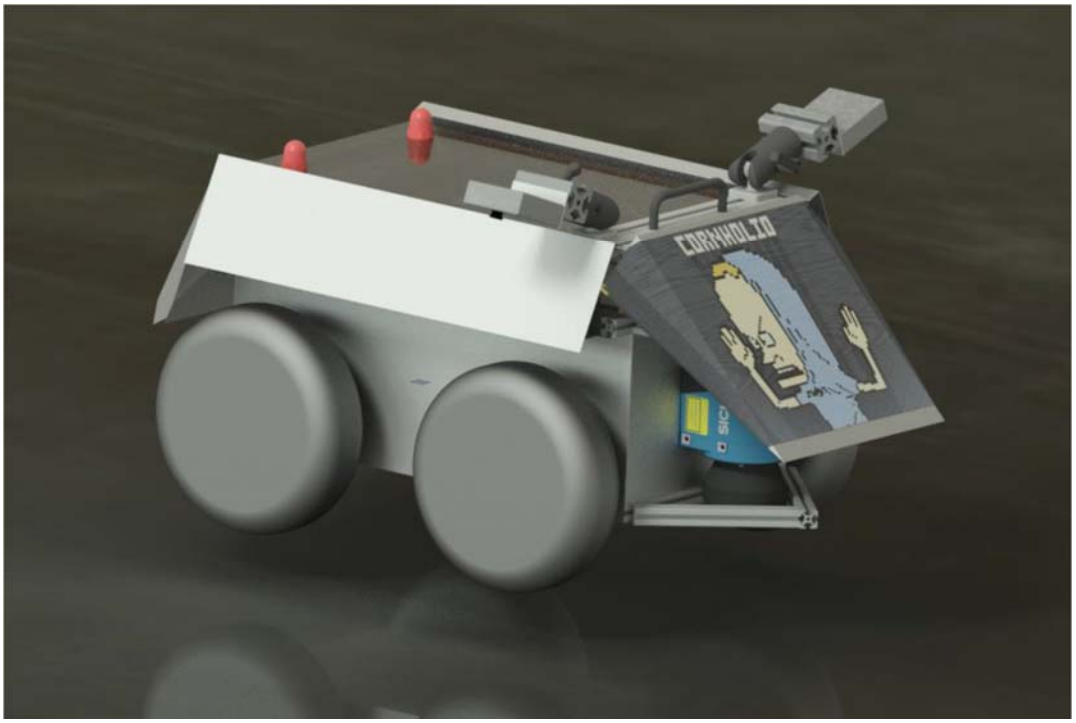


Figure 11: CAD draft of the Robot

''The Great Cornholio'', whose CAD draft can be seen in figure 1, has an overall size of 54x69x44cm. As is typical for robots based on ''Volksbot'' - the robot-platform of the Frauenhofer Institute – Cornholio's case relies on item profiles, making it easy to mount sensors, motors, etc. The profiles and the cover panels are made of aluminium. As a light metal, the usage of aluminium saves weight so the robot will be able to move faster. When the top plate is removed, two claps provide fast access to the fuse-boards and cables.

Two Maxon motors with an epicyclic gearing power the robot with a torque of 15N-m. Each wheel is connected to the motor shaft by a claw coupling. The other wheel is then connected to the first by a chain gear. Separating the drive sections makes it possible to control each side of the robot independent of the other as is typical for skid drive. An advantage of skid drive is the low turn-radius, thus optimizing maneuverability. This behaviour is useful in navigating through the narrow curved rows of the contest environment.

As already mentioned, the robot's case consists of item profiles. Item offers a special ball joint fitting in their profile system that we used to mount our LeanX smartcams. The cameras can be slided on a horizontal line using the clamp lever to change the camera's roll-pitch-jaw ankle. This provides great flexibility for the image processing sensor's field-of-view. The camera can be shifted simply by turning to lever providing flexibility and adaptive capabilities in a changing environment (e.g. from the laboratory to field conditions) within minutes.

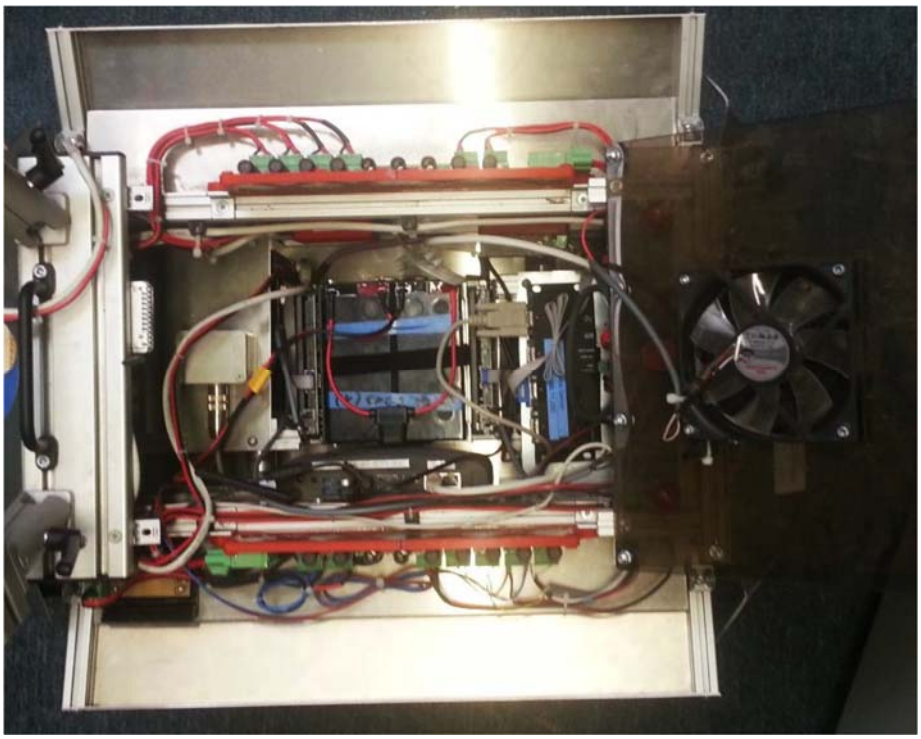


Figure 3: Inner composition of the robot

Figure 2 shows the inner composition of Cornholio. The Wi-Fi access point is fixed on the device server with adhesive hook-and-loop tape which is also connected to the aluminium plate. Just below the plate one of the motors is mounted. If there are any problems with the motor, the plate must be removed without moving the access point and device server. The switches are mounted in an upward position, making the status LEDs observable without interference from other components. The switches are positioned in the front and the back of the robot.

3. Sensors

LeanXcam

For “weed detection”, two LeanX smartcams have been mounted to the front robot pointed toward the ground at approximately 45° away from the front orientation of the robot. The leanXcam is an open-source smartcam that uses a CMOS-Sensor with a resolution of 752x480 pixels and a frame rate of up to 60 frames/s.

Figure 4: LeanXcam

Source: https://github.com/scs/leanXcam/wiki

Laserscanner

For the detection of plants and obstacles there is a Sick Laserscanner LMS100 placed at the front of the robot. The Sick Laserscanner uses the “time of light” method for measuring the distance between the reflecting object and the scanner. The scanner is capable of measuring the surrounding area in an arc of 270° about its vertical center axis.

IMU

The robot uses the Razor IMU (Inertial Measurement Unit) which incorporates 3 accelerometers, 3 gyros and 3 magnetometers. The sensors are used to determine the speed and position of the robot.

Incremental encoder

The motion of the motor itself is measured with an encoder which is integrated into the motor. The information of the encoder and the information from the IMU are used to increase the accuracy of the positioning.

3.1 Hardware

Computer:

- Model: Pokini-i
- Processor: Core i7-3517UE
- RAM: 8GB DDR3 SO-DIMM
- HDD: 120GB SSD
- Bluetooth integrated
- Passive cooling

Figure 5: Pokini I – car PC

Source: http://www.pokini.de/cms/pokini/fileadmin/project/downloads/pokini_i-composing.jpg

The computer is used to control the whole robot. It is connected to all devices by ethernet.

Motor:

- Model: Maxon RE 40
- Nominal voltage: 24 V
- Nominal speed: 6930 rpm
- Nominal torque: 170 mN-m
- Nominal current: 5.77 A
- Stall torque: 2280 mN-m

- Stall current: 75.7 A
- Max. efficiency: 91 %

Figure 6: Maxon motor

Source: http://www.maxonmotor.com/medias/sys_master/8797180919838/RE-40-40-GB-150W-2WE-mTerminal-Detail.jpg

Motor controller:

- Model: Maxon Motor EPOS2 70/10
- supply voltage VCC: 11…70 VDC
- Max. output voltage : 0.9 • VCC
- Max. output current Imax (<1sec): 25 A
- Continuous output current Icont: 10 A
- Switching frequency: 50 kHz
- Max. efficiency: 94%
- Max. speed: 100 000 rpm

Figure 7: Motor controller

Source:http://www.maxonmotor.ch/medias/sys_master/8806425722910/375711_Hardware_Reference_En.pdf

For motor-control there are two integrated motor-controllers used to manage the speed and position of the wheels.

Display:

- Model: EA KIT129J-6LWTP
- Supply voltage: 5 VDC
- Resolution: 128x64
- Integrated touch panel
- Programmable with PC
- Connected by RS-232

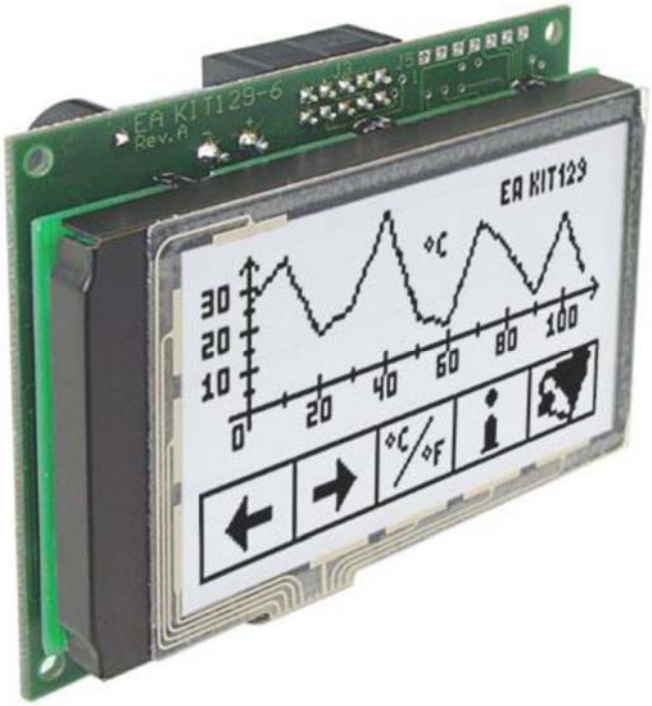


Figure 8: Touch display

Soure: http://www.lcd-module.de/pdf/grafik/kit129-6.pdf

The display with integrated touch panel generates a graphical user interface used to control the robots functions.

Device server:

- Model: EX-6034
- Supply voltage: 5 VDC

- Chip-Set: Samsung S3C4510B
- Data transfer rate Serial: 50 Baud up to 115.2KBaud
- Data transfer rate Ethernet: 10/100Mbps

Figure 9: Device server

Source: http://www.exsys.de/media/files_public/djnxtsiodmuy/ex_6034.pdf

The device server builds a connection for all serial-connected devices to ethernet.

Digital I/O converter:

- Model: EX-6011
- supply voltage: 5 VDC
- Digital Input Lines: 4, CMOS/TTL Compatible
- Digital Output Lines: 4, 2 non-inverted and 2 inverted, CMOS/TTL Compatible
- Data transfer rate Ethernet: 10/100Mbps

Figure 10: Digital I/O converter

Source: http://www.exsys.de/media/files_public/upuwdlpuik/ex_6011.pdf

Relay shield

- Supply voltage: 5 VDC
- Supply current: 150 mA
- Switchable current: 10 A
- Switchable voltage: 250 VAC

Figure 11: Relay shield

Source: http://www.ozhobbies.eu/bauelemente/relais-interface-shield-4x-10a/

The digital I/O converter and relay shield are used to switch loads controlled by ethernet messages.

Power supply

The power supply is based on two 12V lead batteries running in series to provide 24 VDC. They are directly connected to a power supply board that provides three fuse secured voltage levels - 5V, 12V and 24V. The 5V and 12V voltage levels are realized by board intern step-down converters and have a maximum current of 5A. The 24V level is able to provide 20A without conversion. Additionally, there is a buck converter connected to the 24V port of the power supply board that provides additional current to the 12V output port.

Fuse Board

The fuse board connects all electrical components of the robot with their corresponding operating voltages and protects them from exceeding voltage specifications as well. The fuse board supports three different voltages - 5V, 12V and 24V. Every route is fused separately with easily accessible fuse holders for easy exchange.

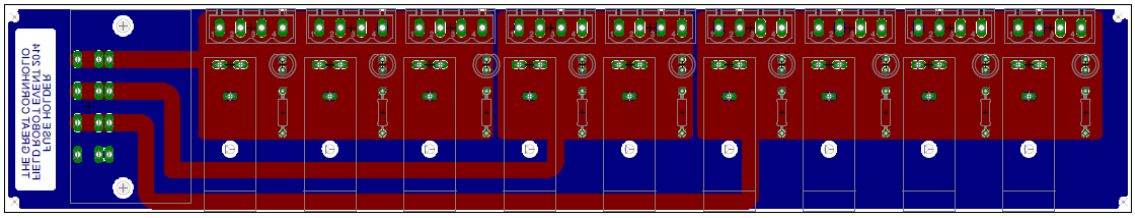


Figure 12: PCB-design of the fuse boards

The function of each fuse is displayed by a red LED. A working fuse bypasses the LED. The bottom side of the fuse board is moulded with epoxy resin. The advantage of this layer is the higher stability and contamination protection of the entire board. The electrical parts are connected with this board through a four-pin connector. This allows a hard-coded pin combination to prevent the interchange of operating voltages.

Figure 12: Fuse boards

3.2 Software and strategy

In order to add modularity to our system, every hardware part of the robot communicates directly or indirectly over TCP/IP protocols. This set up allows to access, test and control the system or any part of it from any computer over a simple wireless router.

The control software:

The core of the software is based on the Robot Operating System (ROS) and makes use of the built-in state-machine functionality to perform the main control procedures. The software initially starts the state-machine server and decides which action is to be carried out depending on the present task and determination of the robot's state. Each action defines a set of different parameters and instructions to be carried out in order to reach a specified goal. Once the end of the action is reached, the action returns completion data and reverts control to the state-machine server. Examples of actions are row navigation, record GPS location, cease motor action, etc.

The navigation algorithm:

The employed navigation algorithm has proven reliable despite, or thanks to, its apparent simplicity. The navigation algorithm reduces the space in front of the robot into an occupancy matrix of 4 rows and 4 columns, although the exact number and dimensions of each cell may change during operation to suit the current environment and task at hand. When the ranges of the LIDAR exceed a given threshold within a cell, the cell is labeled as unavailable and the robot will alter its course accordingly.

Image processing

Image processing was done using the OpenCV (Open Source Computer Vision) library of programming functions and the Eclipse IDE in combination with the LeanX cam hardware. The OpenCV library makes image processing relatively simple which makes for a simple algorithm for the goal at hand. In order to determine the position of a "weed"(golf ball), pixel data is analyzed against HSV values representing the determined range of likely values for the coloration of the golf ball. If the minimum amount of pixel area meets this criteria, the position is considered a match and the GPS data is recorded as the location of a weed.

4. Conclusion

Overall, the robot performed well in the competition. One major limitation in our approach was the amount of field testing performed prior to the event. This factor left us with a significant amount of troubleshooting to be accomplished in the hours leading up to each event. Given more time, the navigation algorithm could have been perfected earlier which would have enabled us to perform significantly better in the basic navigation task, as well as complete the programming for the freestyle competition which we were not well prepared for. The image processing also proved to be difficult due to the impact of the sunlight on the pixel values seen by the camera. Our last minute shade structure that was added to the robot proved to be somewhat effective but could have still been improved. Beyond these, the Great Cornholio proved to be a relatively robust design and capable of performing the required tasks with sufficient amount of competency.

5. References

https://github.com/scs/leanXcam/wiki - last accessed 01.06.14

Trackking

Dini Ras, Jan-Willem Kortlever, Andries van der Meer, Robin Cornellissen, Valentijn van Rooyen, Sebastiaan Dreessen

Wageningen University, Farm Technology Group, Wageningen, The Netherlands

CHASSIS				SENSORS	
WxLxH (cm):	**30x40x40**	Weight (kg):	**10**	☒ Camera	☒ Laser
Model/Make:	**AT-1.0i**	Number of wheels:	**6**	☒ Compass	☒ Odometry
Drivetrain concept:	**Direct drive**	Turning radius (cm):	**0 cm (turn around the axis**	☐ Sonar	☒ Gyroscope
Battery time:	**2h00m**	Rain resistance:	**Splash proof**	☐ IR	☐ Mechanical sensor

Robot software description
The software is build up in LabVIEW

Robot hardware description
Computer: MSI Z87I, Intel Core i7 4770T Compass: Xsens Motion Tracker MTi Laser: Sick Laser scanner LMS111-10100 Batteries: Hacker TopFuel LiPo 20C-ECO-X 5000 mAh Cameras: Microsoft LifeCam HD-5000

Task strategy description
Task 1: The laser scanner scans the rows and with that information we make a preferred route with an intersect and a slope. **Task 2:** Using laser scanner, which recognizes objects. **Task 3:** Using vision (cameras) and comparing colours/shapes. Lights and a patlite are used for signalization and a speaker for a sound signal. We will save the location of the golf balls in a map by using a GPS module. **Task 4 (Cooperation):** We are not going to prepare this before the event starts, if we find a partner during the event to do this task we will enter this part of the FRE **Task 5 (freestyle):** Only if there is time left after preparing the other tasks

TRACTACUS

Eray ÖNLER, Ferhat SERVİ, Cansu ÖZTÜRK, Ceylan OMAZ, Buse ÇAKIR, Yeşim TÜFEKÇİ, Bahar DİKEN, İlker Hüseyin ÇELEN, Erdal KILIÇ

Namik Kemal University, Biosystem Engineering Department, Tekirdağ, Turkey

1. Introduction

Tractacus team was established in 2011 by a group of students at biosytem engineering department. The team's aim is desinging and building simple robots with low cost sensors and chassis. Third version of the robot have wooden chassis and ultrasonic/odometry/gyroscope sensors for navigation.

The drive motors are separated for each of 4 wheels and all wheel can be controlled independently.

2. Mechanics

The chassis is made by wooden. This is strong material and easy to construct, also very light weight. Robot have differential steering. So all wheels have individual motor. DC 12 V, 200 rpm RUNNER motors were used at robot. These motors can give high torque up to 12 kg/cm.

Figure 1: DC motor

Figure 2: Tractacus view

3. Sensors

Ultrasonic Sensor: The ordinary sensors in low budget robots are ultrasonic rangers. In this robot 5 pieces of HC-SR04 ultrasonic sensor were used. 4 of them used in each corner of the robot, and the last sensor were installed in front, of robot.

Figure 3: ultrasonic sensor

Encoder: Two cannel magnetic encoder were used at each wheel for odometry.

Figure 4: odometry sensor

Gyroscope: Gyroscope sensor with 300°/s sensitivity were used for measuring heading of the robot.

Figure 5: gyroscope sensor

Camera: A4Tech 1080p webcam will be used for golf ball detection at task 3.

Figure 6: webcam

3.1 Hardware

Arduino Mega were used for controller board .

Figure 7: arduino mega

Pololu Dual MC33926 motor driver were used for driving motors.

Figure 8: motor driver

3.2 Software and strategy

Row following: The robot needs to navigate in two distinct situations: between the rows, and on the headland . While between the rows, the robot follows the line that lies in the middle of the path between two rows of maize. While on the headland, the robot follows a line that parallels the imaginary line that can be drawn between the ends of the maize rows and that lies at a given distance from the ends of the maize rows. Of course, these two situations reduce to one, namely following a straight line, where the variables to be controlled are the distance from the control point to the line that is being followed, and the angle between the robot's current path and the line that is being followed.

Headland following: The robot continues following the rows until it has cleared the rows by a preset distance. Then it comes to a full stop, makes an on-the-spot turn to position the robot parallel to the new path, and follows the new path. When it has reached the middle of the new row, it comes again to a full stop, makes an on-the-spot turn, and starts following the rows again.

Obstacle: If an obstacle is detected during the first few meters of a new row, the robot backs out and returns to its last position outside the row, and then travels on the headland to the next row.

Golf Ball Detection: The web camera is directed at a right angle to the ground so there is no need to geometrically correct the image. RGB filter was used to separate ball with background.

4. Conclusion

A good field robot has to be able to adapt to any kind of field, whether it is in good condition or not. To be good in row driving in any conditions, multiple sensors have to be used as one may work better in some conditions and some in the other. Tractacus was equipped only with ultrasonic rangers, gyroscope and odometry sensors.

5. References

http://www.robotistan.com/2-Kanalli-Manyetik-Enkoder-Magnetic-Encoder,PR-1166.html

http://www.robitshop.com/LPY503AL-2-eksen-jiroskop-Gyro-Breakout-Board-LPY503AL-Dual-30s,PR-1919.html

http://www.robitshop.com/Arduino-Mega-2560-R3-Yeni-Versiyon,PR-1735.html

http://www.robotistan.com/HC-SR04-Ultrasonik-Mesafe-Sensoru,PR-1473.html

http://www.robotistan.com/37mm-Runner-200-Rpm-Reduktorlu-DC-Motor,PR-901.html

FinnFerno

Johan Vepsäläinen[1], Aki Laakso[2], Lauri Andler[1], Jussi Enroos[1], Juha Hietaoja[3], Genki Ikeda[2], Tapani Jokiniemi[3], Aku Korhonen[2], Jussi Raunio[3], Iiro Salminen[2], Tatu Tolonen[1], Timo Oksanen[1*], Jari Kostamo[2*], Matti Pastell[3]

[1] *Aalto University, Department of Automation and Systems Techonology, Aalto, Finland*
[2] *Aalto University, Department of Engineering Design and Production, Aalto, Finland*
[3] *University of Helsinki, Department of Agricultural Sciences, Helsinki, Finland*

1. Introduction

The Field Robot project is a joint venture of 11 students of Master and PhD level from Aalto University and University of Helsinki. The project brings together the expertise of computer sciences and control systems, mechanical engineering and mechatronics as well as agricultural science and technology to design and build a fully autonomous agricultural robot.

This project has been organised between Aalto University departments of Automation and Systems Technology and of Engineering Design and Production and the department of Agricultural Sciences from the University of Finland. In previous years it has been determined that the incorporation of several disciplines working towards a common goal provides the necessary know-how to implement a completely new robot in consecutive years. For our team this has enabled the design of a much more intricate and complex system.

The Field Robot project aims at the implementation of an autonomous robot that is entered to the Field Robot Event of that year. In the 2014 Event Tasks 1 and 2 deal with navigation performance in a corn field. Task 3 incorporates a weed detection and geolocation subtask to the navigation in a corn field. In Task 4 two robots are assigned as cooperation partners with the specifics of the task left to the partners. Task 5 is a freestyle task, where each team presents an agriculturally significant performance that they have assigned themselves.

Previous robots were extensively studied and well-performed features are used as a starting point for the new robot. For the 2014 Field Robot Event the team decided to improve on several aspects of previous Field Robots but still build a completely new robot from scratch. Primary focus has been on further development of the *axle module* used in previous years, specifically in designing a fully independent steering and drivetrain for each wheel. In terms of computer hardware there is a major increase in processing capability, compared to previous robots.

For the team members this project is also meant to act as a cross-discipline learning experience and provide an alternative perspective to a traditional academic project. In order maximise efficiency the group was divided to three working groups: The automation design group, the mechanical design group and the trailer design group. Design for the main robot of FinnFerno was given to the automation and mechanical design groups. The learning goal was to provide the machining and mechanical design experience to the automation students, and programming and electrical design experience to the mechatronics students. The agricultural technology students were given responsibility of the trailer system that is used in Task 5 of the Field Robot Event.

2. Mechanics

Mechanical construction of FinnFerno is primarily made of aluminium. Excluding the drive and steering motors and gearbox, all of the mechanical construction in the wheel modules is custom designed and assembled. Most sheet aluminium components were laser-cut commercially by the Finnish company *Laserle Oy*. Components that could not be made using sheet metal were machined at Aalto University premises. These components include steering transmission wheels and certain parts of the gearbox input and output shafts.

Axle Modules

The mechanical design for the robot was started in October 2013 and finished in February 2014. This included design for the axle and wheel modules as well as the chassis. The novel design of the axle modules was a major factor in the relatively lengthy design process

The axle modules are independent, modular components that contain the motors, drivetrain and steering motors, and the necessary electronics for controlling and monitoring those motors. In total three axle modules were prepared, two of which are used in the robot and one kept as a spare in case of motor burnouts or other malfunctions.

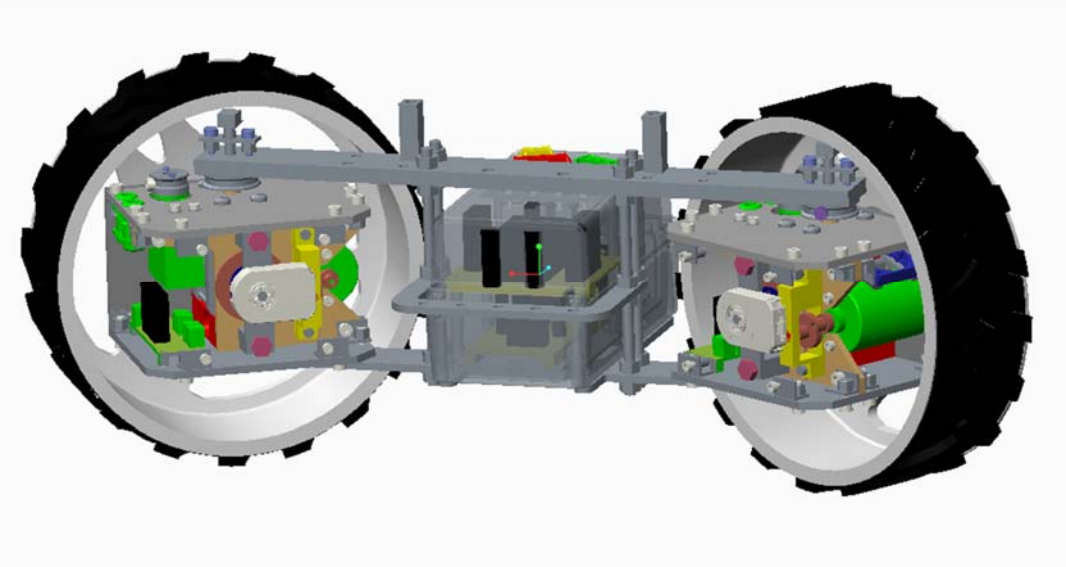


Figure 3.12: *CAD of the axle module. Middle module houses electronics, wheel modules the motors, transmission and steering servo*

Wheel Modules

Each axle module has two identical wheel modules and one central electronics module. The wheel modules house the Banebots RS540 12V DC motor, Banebots P60 Gearbox with a 26:1 transmission ratio and HK15298B servos used in the steering mechanism. Additionally, the wheel module uses a potentiometer to measure wheel turning angle and a thermistor for measuring the motor temperature. Each wheel of the robot has an AMT102-V magnetic encoder on the transmission input shaft with a resolution of 2048 ticks per rotation. The odometer measurement is used for the kinematic model of the robot. The mounting of the encoder is shown in Figure 3.13

The wheel modules are built to be relatively simple to assemble and service. This is imperative since the original wheel module design allows for steering angles of up to 60 degrees in each direction, leaving only very limited space for the motor, transmission and steering servo.

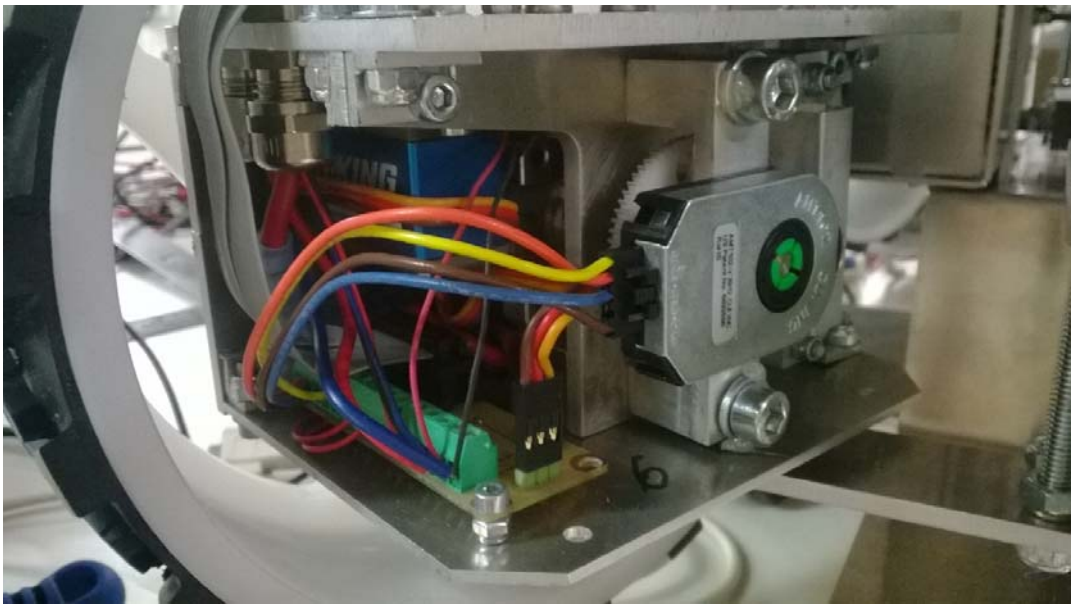


Figure 3.13: *Close-up of the wheel module, demonstrating the limited size inside*

Electronics Module

The central electronics module houses two custom made H-bridges and a custom made motor controller board. The H-bridges are used to control the drive motor voltage with PWM, regulate supply voltage for the steering servo and measure the current flow through the drive motor and steering servo. Since the steering servo can consume significant amounts of energy, two 6V regulators are used to power it. In order to compensate for varying power demand in the steering and drive motors, the power lines are stabilized with 1 *mF* capacitors.

The motor controller board interfaces the command signals from the control software to the motors and steering. Additionally it relays the status of both wheel modules in the axle module to the control software. The motor controller has an AT90CAN128 microcontroller, mounted in a Crumb128-CAN development board (see 0). The microcontroller is used for PWM generation, data acquisition and communication with the rest of the system.

PWM generation for the control of the servos and the wheel motors is done in the motor controller. During development it was noted there was a risk of breaking one or more of the gears in the drivetrain when it is put under mechanical load, such as accelerating from a standstill or stopping suddenly. A PD controller that is used in every drive mode was devised to mitigate this risk. The controller limits the rate of change in the motor output based on the difference of the current output and the requested output. In addition to this a PID controller was implemented for the closed loop driving mode that was used for autonomous operation.

Tuning of the PID controller proved straightforward, with a satisfactory balance of stability and responsiveness achieved without the use of the derivative term. The closed loop controller was therefore just a PI controller, but the complete control loop inherited the derivative term from the drivetrain PD controller to further stabilize the motor output.

For the servos a PD controller, similar to the drivetrain PD controller, was used to protect the steering linkage from sudden actuations. Although at first the need for a controller for the servos was thought to be unnecessary since the servos had very little persistent error, the controller proved to be a major factor in keeping the servos and the steering linkage from breaking under autonomous drive.

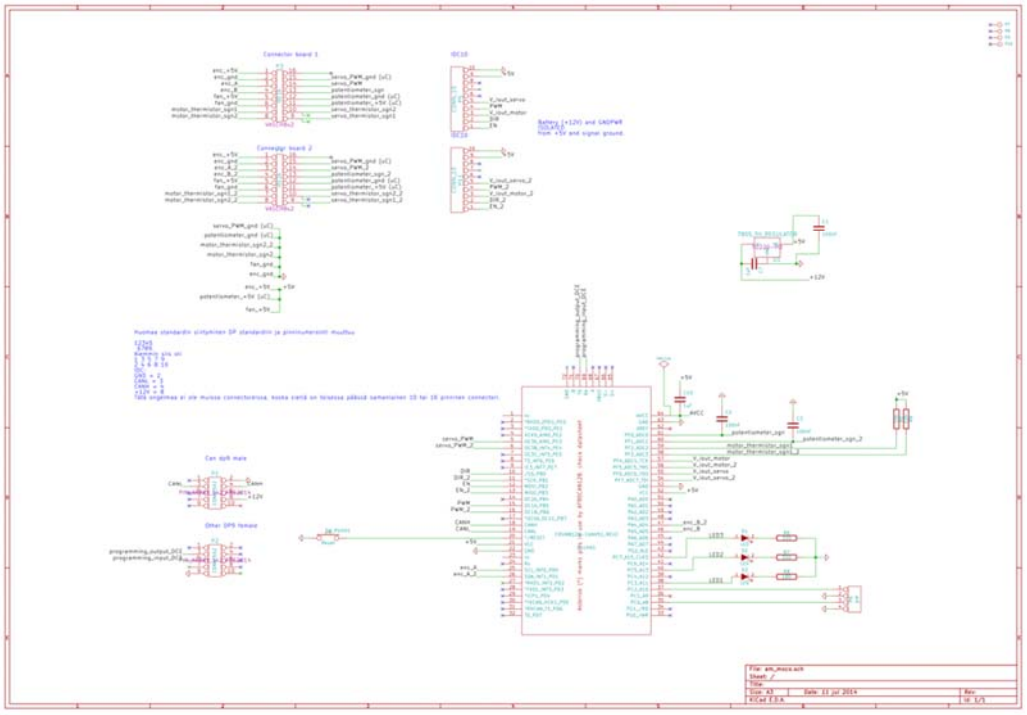


Figure 3.14: *Axle module motor controller schematic shows all the signal inputs and outputs used in an axle module*

Chassis, Wheels and Cover

The chassis is custom designed to maximize mechanical stability and minimize unsprung weight. A majority of components in the chassis construction are laser-cut aluminium plates, assembled by hand. The electronics of the robot system were mounted on a separate polycarbonate plate, with the intent of separating the electrical system of the frame of the robot, but also to allow easy access to the electronics separately of the robot frame.

Previous robots and their suspension and chassis design were studied and referenced in order to save time in the design phase. The robot has a “twisting torso” -type

balancing system, which has been used in several previous Finnish field robots. Especially the 2011 robot, “Cornivore” was used as a reference because of its good performance. The idea behind the balancing system is that it allows both axles to twist freely around the robot’s longitudinal axis, but always in opposite directions. This allows the load on each wheel to stay the same and keeps the robot’s main frame close to horizontal at all times.

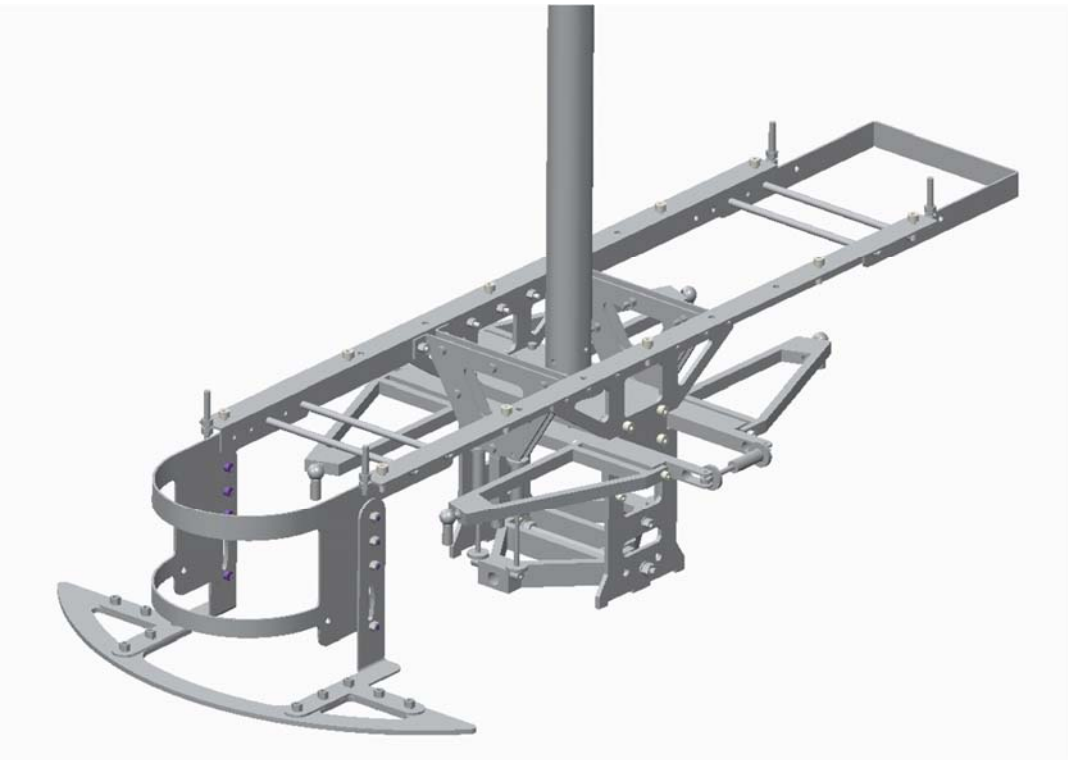


Figure 3.15: *Robot's structural frame. The front has been designed with a bumper to protect the LMS100's laser ranging mechanism from impact and with additional protection for the housing of the sensor. The upper protection is also used for carrying the robot*

The balancing system has been suspended both longitudinally and laterally with shock absorbers and springs. The shock absorbers were ordered from RC4WD and the springs were bought separately from jouset.com according to the robot’s dimensions and estimated weight. The axle modules are connected to the suspension system via three ball-socket joints, which allow them to rotate and move vertically with respect to the robot body.

By removing the joints, the axle modules could be detached and reattached in less than 15 minutes, although their locations were somewhat difficult to reach with a hex key. A lot of laser cut components were utilized in the suspension system, although some parts were self made, including 6 triangle pieces that were cut from aluminium plates with a wire EDM machine. The suspension system worked well during testing the competition without any mechanical problems.

The wheels have a diameter of ~20 cm, which has been determined to be the best size in previous years. The rims are machined using a lathe and an NC milling machine from a solid piece of white PE. Rugged rubber tires are added for improved traction. The tires are partially secured by the ridges on the inner and outer edges of the rims.

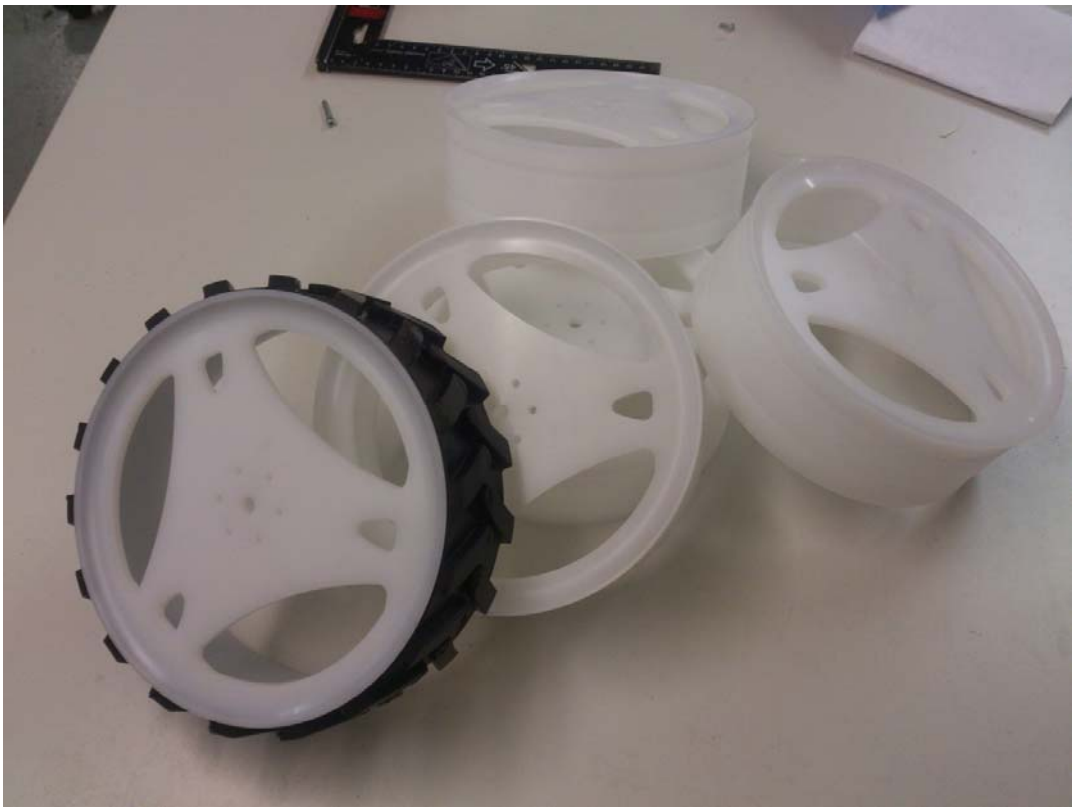


Figure 3.16: *The wheels were manufactured from solid pieces of plastic, making them quite expensive to make*

The robot's cover was made from vacuum molded plastic sheet, ordered from *Merocap Oy*. The mold was manufactured with a novel technique, in which the model was 3D printed from quartz according to a solid CAD-model and hardened with epoxy. To achieve an even surface, the mold was finished with sandpaper, although at some places the surface epoxy layer wore off and the brittle sand texture started to rustle as can be seen in Figure 3.17.The sponsor and logo stickers were put in place and finally a few coats of lacquer were sprayed on to finish the cover.

Figure 3.17: *Final cover and 3D printed mold*

3. Sensors

The robot has several different types of sensors to provide options and robustness in the field. The primary sensors for Tasks 1 and 2 are the ultrasound and infrared sensors as well as the laser range scanner. Twin cameras were used for Task 3 for the weed detection, but the cameras also provided positioning data for row navigation.

Infrared and Ultrasound

Sensors

The robot has infrared and ultrasound sensors in each corner of the robot. Since these sensors have been proven reliable and accurate enough in past years, they have been integrated to a single sensor module that can be easily housed and mounted to the robot. This module template was created by the instructors.

The module has a SRF04 ultrasound ranging unit and a GP2D120 infrared ranging unit that are controlled by an Atmel ATMega88 microcontroller. The collected data is transmitted via RS-485 transceiver, also integrated to the module. Each “RangePack” (Ryynänen & al, 2011) module communicates with a custom-made hub module using the RS485 communication protocol.

Figure 3.18: *The Rangepack module has an infrared sensor (left side) and ultrasonic sensor (right side)*

Sensor Hub

The RangePack Hub is designed based on communicational requirements. Each individual RangePack sensor pack is connected to the hub with a 6-pin flat cable. The connection provides the RS-485 communication line and a 5V power supply to each sensor. The hub acts as the master in the RS-485 network, polling the sensors at a given interval. The sensor data is multiplexed to two individual CAN messages, front and rear sensors respectively.

Data processing and communication is done by a Crumb128-CAN module on the hub PCB. The hub was tested to perform without fault all the way down to a polling interval of 30 ms for all four sensors. At and below this threshold the hub began to exhibit dropped messages and other abnormal behaviour.

Figure 3.19: *Rangepack Hub*

In order to limit the load on the Crumb's regulator due to the 4 RangePack's power draw and voltage losses over the cabling, the Crumb module has a dedicated 5V regulator. The RS-485 bus was also equipped with signal filtering, done by connecting the bus to either the 5V power supply or the ground through a 560 Ohm resistor. This option was not needed, but in a very noisy environment this filtering could have provided better signal characteristics.

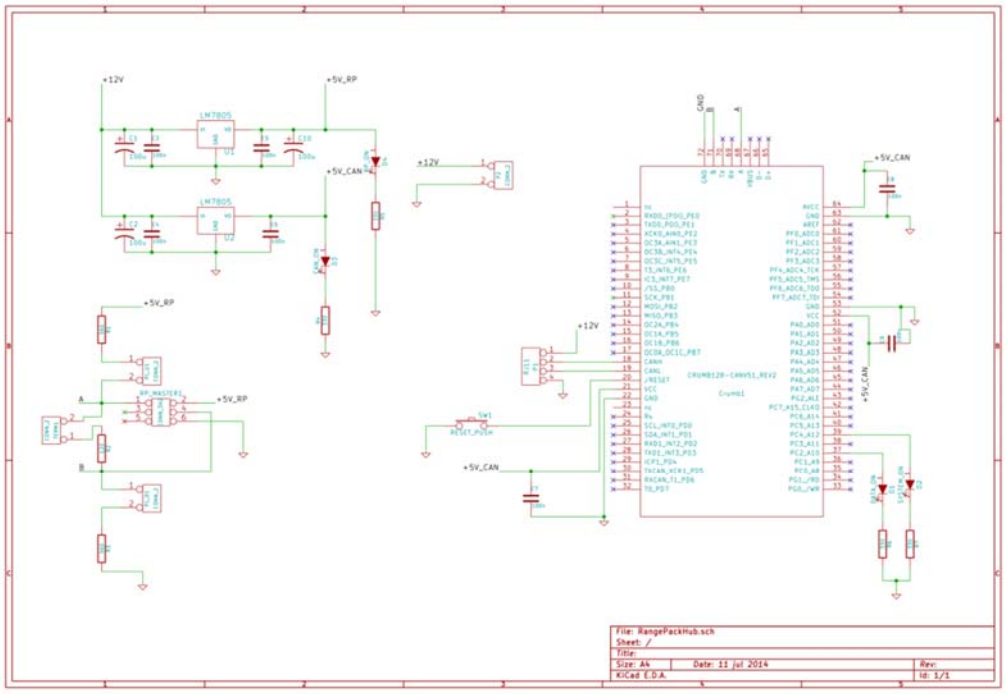
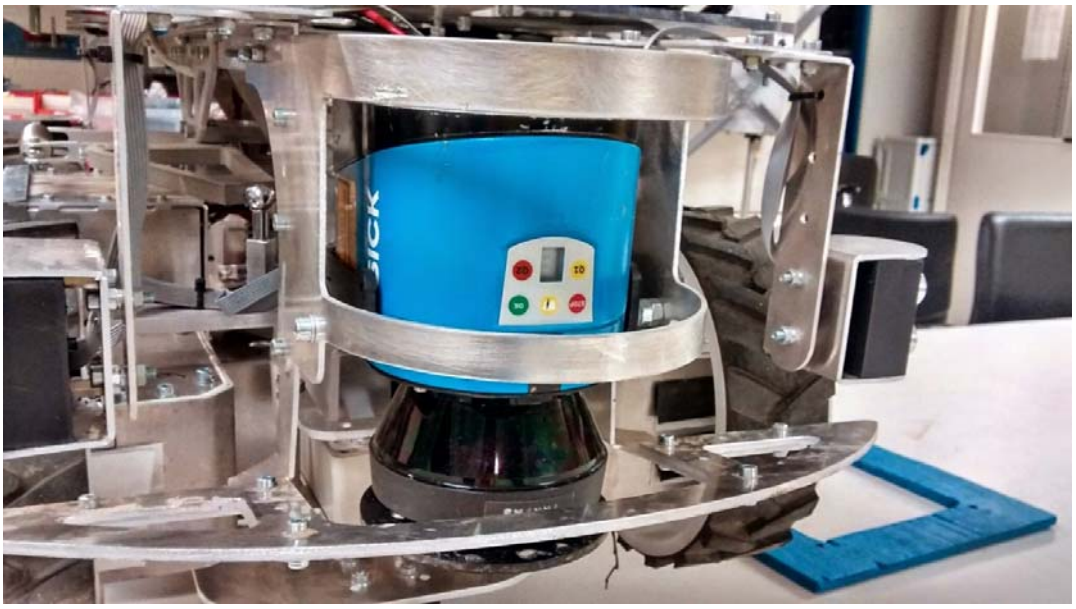


Figure 3.20: *Schematic of the Rangepack Hub. The Crumb has a dedicated regulator to compensate for the power losses of the Rangepacks.*

Laser

The robot has a SICK LMS100 laser rangefinder unit that is mounted in the front of the robot. The sensor has a 270 degree scanning sector, with a 0.5 degree resolution. The laser sensor functions at 25 *Hz*. The laser is used for row end detection, obstacle detection and positioning in the rows and headlands.



Cameras

In order to acquire higher resolution images and have depth-perception capabilities, it was decided that two Gigabit Ethernet industrial cameras are going to be used. The cameras are mounted along the same lateral axis at the top of the camera mast, at a changeable height of 105-135 cm from the ground. The cameras were used in Task 3 for detecting the weed plants, and were used for positioning in the rows and headlands.

Figure 3.21: *Pixelink Gigabit camera*

3.1 Hardware

Microcontrollers

Crumb128-CAN with Atmel AT90CAN128

The principal microcontroller used in this project was the Atmel AT90CAN128, which was integrated to a commercial development board, the Crumb128-CAN (Chip45, 2014). The Crumb128-CAN provides an external 16 *MHz* oscillator, CAN, RS-232 and RS-485 transceivers on a modular board.

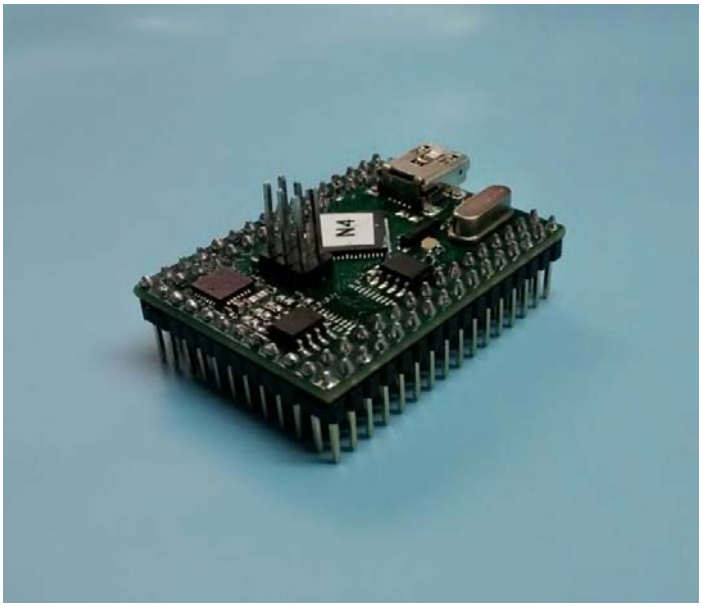


Figure 3.22: *Crumb128-CAN board with headers. This board is equipped with RS-485 transceiver (bottom right IC) but no RS-232.*

Modularity proved a vital asset, as a few Crumbs started malfunction very late in the project but were easily replaced with working modules.

Atmel ATMega88

The Atmel AtMega88 is a very versatile microcontroller; it is affordable and provides a wide range of functionalities, such as, most importantly for this project, an 8-channel, 10-bit A/D converter and connections for an RS-485 transceiver.

Onboard Computers

FinnFerno is equipped with 2 computers, as it has been found in previous robots that more advanced algorithms cannot be run on a single computer.

For navigation purposes an eBox 3350mx computer running Windows CE has been used for several years successfully, and is used again. It has a 1 *GHz* x86 processor and 512MB DDR2 RAM. For positioning algorithms and machine vision applications several

ULV computers, such as FitPCs, have been tried but have been proved underpowered. Therefore a more powerful computer was selected.

The Intel NUC DC53427HYE has an i5 processor, clocked at 1.8 *GHz* and 4 *GB* of DDR3 RAM. The NUC is running Windows 7 32-bit to provide support for a number of modern processor and power management features.

Figure 3.23: *eBox and NUC PCs. The eBox is passively cooled while the NUC has a fan to keep the processor cooled.*

Communication

The computers, both onboard and remote, communicate via UDP over Ethernet using a self-implemented binary serialisation protocol. The twin cameras and the LMS100 are also connected over Ethernet, but use proprietary manufacturer specific communication protocols.

For communication with sensors and axle modules, as well as the trailer, a CAN bus implementation is used. The computers are connected to this fieldbus with Kvaser Leaf Light HS adapters, while the Crumb-128 microcontrollers are equipped with CAN transceivers as standard. CAN communication is routed through a custom CAN hub, which can also provide power to peripherals using the single communication cable.

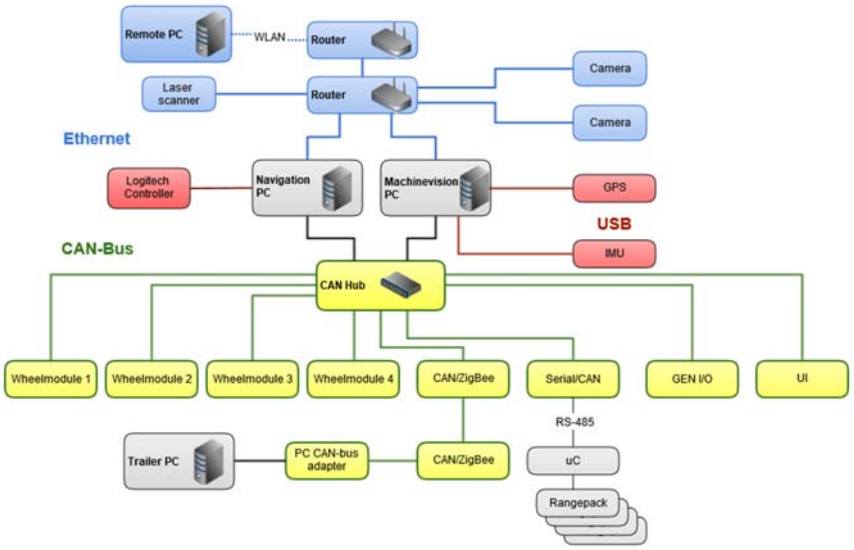


Figure 3.24: *Communication in the robot. Blue lines are Ethernet, Red is USB and Yellow is CAN. Other protocols defined in the diagram.*

CAN Bus and Hub

The CAN bus was selected as the communication method for sensors for two reasons: Firstly, by using a centralized communication bus it was possible monitor almost all sensors of the robot with commercial tools, such a Kvaser Leaf Light HS CAN bus analyzer and CANTrace, thus allowing efficient unit testing of separate modules.

Secondly, with the CAN bus we could implement an ISOBUS-inspired communication protocol, bringing an interesting aspect to the project. Our CAN bus implementation uses the CAN 2.0b standard, with support for the extended 29-bit identifier. For simplicity we only used broadcasted messages rather than addressed messages, as we could use the Group Extension to filter messages to different nodes.

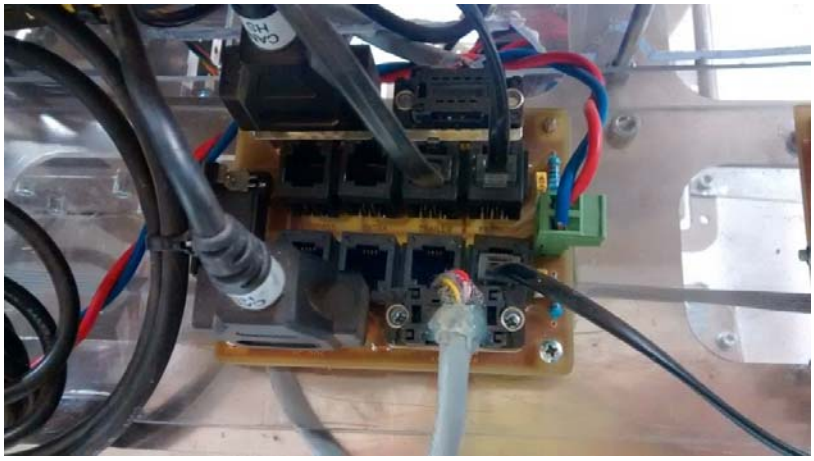


Figure 3.25: *Powered CAN Hub PCB. Peripherals are connected with RJ-11 while computers and axles use DB-9*

The peripherals are connected using two types of connectors, DB-9 and RJ-11. The DB-9 connectors were used for axle modules and the Kvaser modules due to their robustness, and because the Kvaser modules are equipped as standard with them. RJ-11 connectors were used for all other peripherals in order to allow for rapid installation and removal of devices. Both types of connectors, excluding the Kvaser connections, carry CAN H and L channels as well as +12 *V* and GND lines

Ethernet and Wi-Fi

Figure 3.26: *Twin D-Link DGS-105 Ethernet Switches*

The FinnFerno robot is equipped with two D-Link DGS-105 Gigabit Ethernet switches. This model was selected because it was determined that four LAN ports would not accommodate all connections of the robot and a Gigabit link was required for the twin cameras.

An 8-port switch was considered, but since one DGS-105 was already in the inventory, it was decided that two identical switches would be used. This particular model also has a rather compact design, making it very well suited for our robot. An Asus portable Wi-Fi adapter is used to provide an access point to the robot's network.

Figure 3.27: *The ASUS adapter that was used to provide an access point*

Power, Input and Output

Batteries

The robot has four *Turnigy* Li-Po batteries, rated at 11.1 *V* and 5000 *mAh.* At the beginning of the project it was decided that four batteries would be necessary to provide sufficient runtime for the robot on a single charge. In the final design both axle modules have their independent batteries, and the computing system has two power circuits with a number of different configurations.

By leaving free power terminals on the power distribution board it is possible to change the connections of every component between the two batteries used for electronics. Through testing it was found that the NUC computer would draw such significant amounts of power that it could not be connected to the same battery as the LMS100 laser scanner. In the end, one battery provided power for the eBox computer, the CAN hub, the LMS100, the General I/O board and the Ethernet switches, while the other powered the NUC and the cameras. In this arrangement both batteries would run low at approximately the same time.

Battery Protection

Since lithium batteries can provide very high currents if short-circuited, it was determined that the batteries would be connected to a fuse-protected cut-off circuit. The cut-off circuit is also used to protect the batteries from discharging below the breakdown voltage.

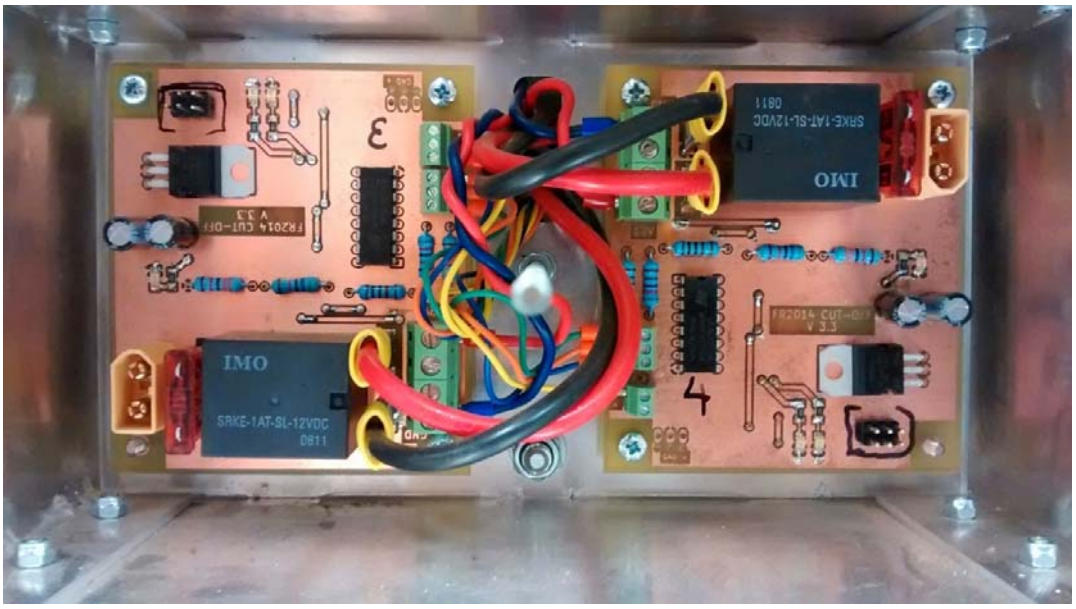


Figure 3.28: *Cut-off circuits mounted to the battery box.*

The cut-off circuits are mounted in the battery boxes, so their size was one of the most important design criteria. Although the PCB design had to be revised a number of times, the final revision was deemed capable of handling the expected currents of the motors. The cut-offs were tested with very high currents using a 55 *W* lamp matrix and they were found to withstand continuous currents of 24 *A* for up to 3 minutes, at which point the battery voltage dropped below 9.5 *V.*

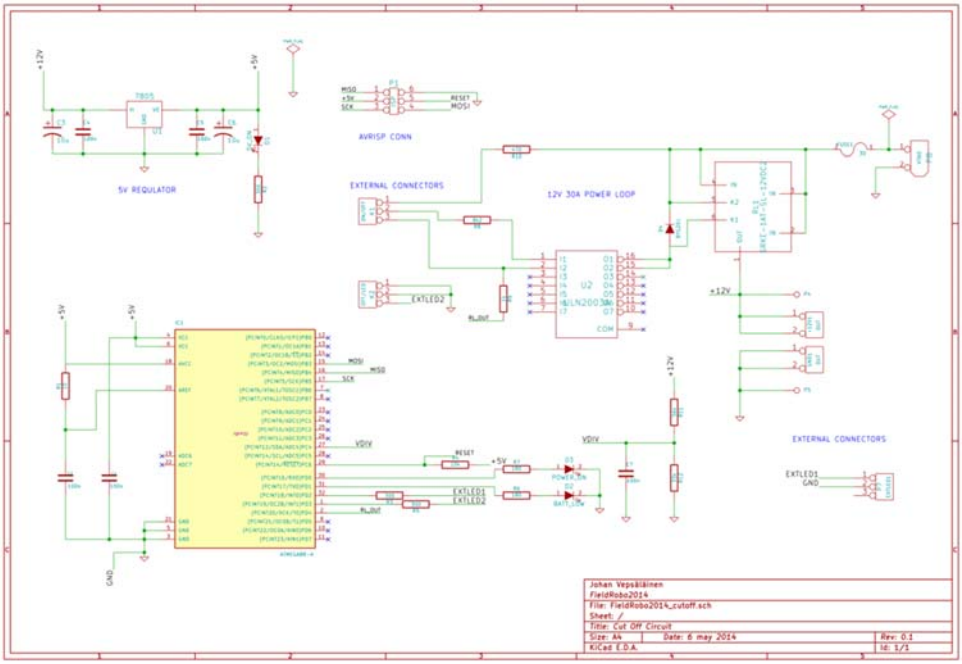


Figure 3.29: *Schematic of the cut-off circuit.*

Even after such high currents, the circuit would only reach an average temperature of40 ^{0}C, with a maximum temperature of 58 ^{0}C measured in the fuse, rated at 40 *A*. The measurement was done using a FLIR infrared camera. The heated fuse is in the upper right corner of the right side image in Figure 3.30.

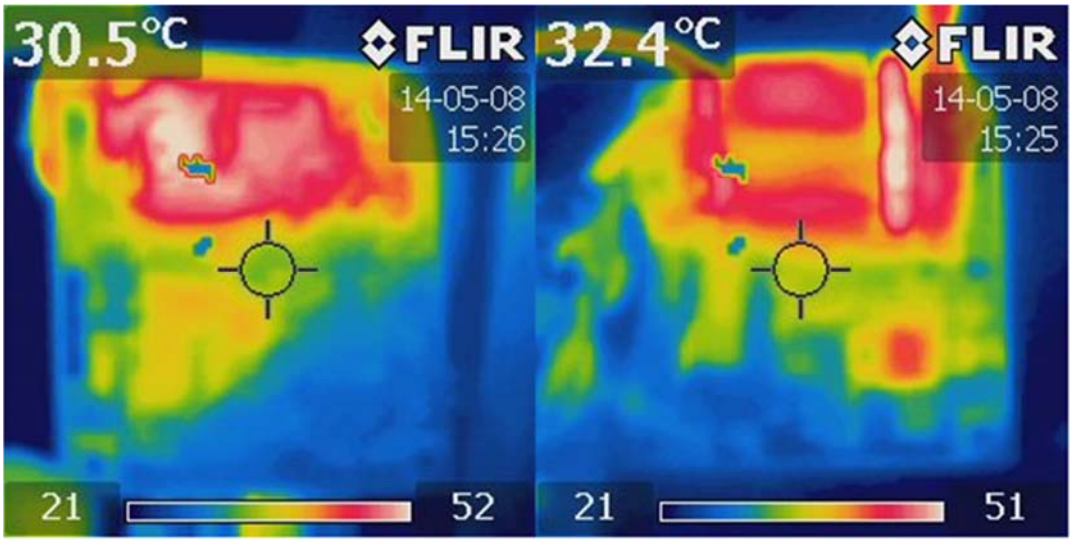


Figure 3.30: *Back of the PCB on the left, front on the right. The bright spot in both images is the area around the fuse. The fuse heats up significantly, but this is the expected result under the test conditions.*

Power Distribution

The power output of each battery is re-distributed in the power distribution board (PDB). The two electronics batteries are further divided into subsystems with individual fuses. The motor batteries were not set up with fuses in the board, as through the PDB these batteries are used only for the camera stepper motor. The wheel motors are connected to their respective batteries directly. For the schematic in Figure 3.31 these fuses were added as a possible point of improvement.

Another reason for bringing all four batteries to the PDB is the possibility to measure the voltage of each battery. The voltage measurement is done on a separate PCB but the voltages are lowered from the 12 *V* range to 5 *V* on the PDB.

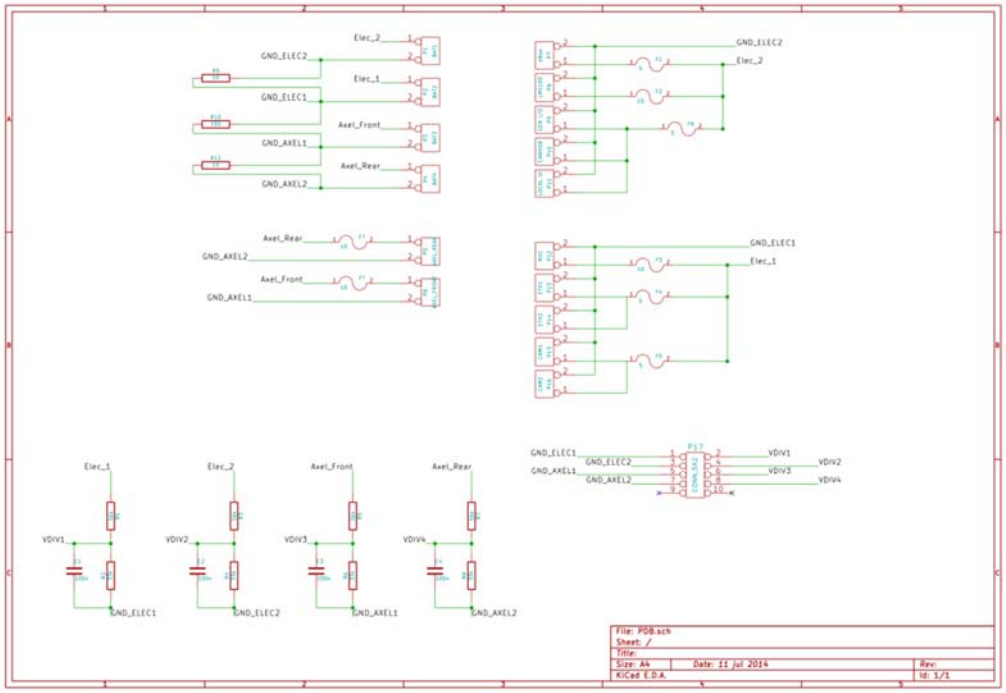


Figure 3.31: *Power Distribution Board schematic*

General Input / Output

The General Input / Output board is used for functions which could not be integrated to other PCBs. It is used to generate the control signal for the camera stepper motor, controlling the buzzer and measuring the battery voltages. As an additional feature it is possible to choose between an indoor and outdoor buzzer.

In addition to these predetermined functions, several input and output ports were left free for other functions, such as signalling lights.

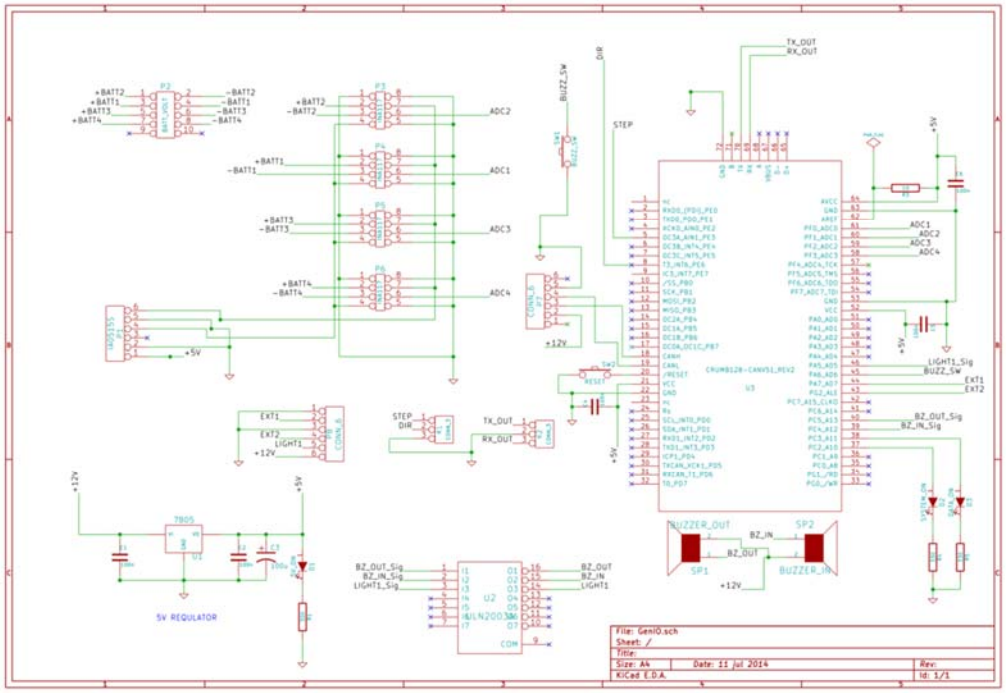


Figure 3.32: *General I/O board schematic. This board is used for auxiliary features of the robot*

Local User Interface

The Local User Interface is used to control the robot's basic functionalities, such as starting and stopping the robot, viewing and editing the turning sequence and ending the main process on the eBox computer. Additionally the logs from the system and detected ball positions can be closed and saved with the Local UI.

The UI design is largely influenced by some of the previous Field robot's user interfaces, namely the EasyWheels (Kemppainen & al, 2009) and Cornivore (Ryynänen & al, 2011) robots. The design principle was to provide easy access to critical functions, with clear indication of each button's functions. Starting from the bottom of the UI panel, also shown in Figure 3.33, there are the main power switches and buttons as well as the emergency stop button in the middle. The left power switch controls the batteries on the left side of the robot and the right side switch controls the batteries on the right side, powering the axle modules and electronics respectively. The rocker switches are used to cut power from the batteries, while the red push buttons turn on either subsystem.

The four push buttons around the LCD screen are used to control the robot during standard operation. The PREV and NEXT buttons are used to display and change the current turn of the turning sequence. The VIEW buttons changes the view on the LCD

and the MODE button is used to alter between row driving and headland turning modes.

There are also a number of button combinations available. By pressing the PREV, VIEW and MODE buttons at the same time, the main program on the eBox is shutdown. This also saves the current log file, containing the raw and SI measurement data from sensors, as well as the SI and raw control signals. Another similar function is mapped to the simultaneous press of the VIEW and NEXT buttons, which writes the current ball recognition data to a text file.

Figure 3.33: *Local User Interface. The plate is mounted on tall threaded rods that are attached to the frame of the robot*

The UI has a number of LEDs which let the user know about the current status of the robot. On the top row, the left- and right-most blue LEDs signify the direction of the next turn and the number of rows to be skipped during the next turn. The three LEDs in the middle of the top row let the user know whether any of the four batteries are running low, divided to motor and electronics batteries respectively. The “System On” LED will blink periodically when the eBox software is running, and will stop doing so if the connection is interrupted or the program is stopped. The lower row of LEDs

signifies the algorithm's status. These LEDs show whether the robot is turning in the headland, reversing or stopped respectively.

The design of the UI was found to be very successful. It provided a simple overview of the system and left very little chance of pushing buttons unintentionally. The extra space on the UI came into good use when it was discovered that, due to the design of the Cut-off circuit, a single rocker switch would not work for two cut-offs as both the ON and OFF switch. Therefore additional push buttons had to installed adjacent to the rocker switches, which was a very minor issue, largely due to the fact that there was plenty of space to set them. In the final configuration the red push button turns the power on and the rocker switch turns it off. For future reference, this could be avoided by using power switches with four dedicated connection channels, instead of the two as in our switches.

3.2 Software and strategy

Software for the robot is partially implemented in Visual Studio as C# code, especially for functionalities such as CAN communication, machine vision, remote UI and parameter handling.

Algorithms on the other hand are developed in Matlab, where they can be more easily tested and much more comprehensive mathematics capabilities are provided as standard. The algorithm models in Matlab are then generated to C-code and a platform invoke interface is used to communicate between the Matlab algorithms and the C# backbone software.

Initially it was decided that for algorithms there can be several alternatives developed, of which the best one can be used. The algorithms are referred to with an alphabetic system where A represents the simplest algorithm and Z represents the most advanced algorithms. In the end only the A-algorithms were implemented with proper synthetic testing. For both positioning and navigation additional algorithm were developed in the very end of the project and even during the Field Robot Event. These were only tested on the fly, but they did work to some extent.

Internal communication

The main software on the robot was divided between the eBox and NUC computers. The positioning algorithms and machine vision was run on the NUC, as it provided more processing power. The navigation algorithm, parameter handling, and remote communication were run on the eBox. Lower level software was run on the microcontrollers.

The two onboard computers used UDP communication to send messages between each other. The UDP packets were binary serialized C# structs. TCP communication

could have been also used to gain error-checking and delivery validation. However, since UDP communication was used in the previous robots and had proven to work reliably, there was no need to update to TCP.

The UDP messages were mainly positioning data from the positioning algorithms and the machine vision algorithm. Thus most of the UDP traffic was from NUC to eBox. Some general messages and parameter messages were also in use.

Parameter handling

All the tuneable parameters were saved on three separate XML files on the onboard eBox computer. The parameters were divided into three categories based on how often they needed to be changed. The p1 parameters were to be changed only when the robot was rebooted. These included e.g. the current order of axle modules on the robot. The p2 parameters were common to all the competition tasks, e.g. positioning parameters. Finally the p3 parameters were to be tuned separately for each competition task. These would include, for example, the algorithms currently in use or target driving speed.

The parameters were mostly determined in Matlab. As the Matlab models are code-generated into C, the marked parameters are automatically put into a struct. If there were changes in the parameters (i.e. parameters were added, removed, or renamed) after the last code generation, the changed parameters were to be copy-pasted into structs in the C# code. This was the only required manual step if there were changes in the parameters.

The handling of the parameters in code and between the C and C# structs was done using the reflection feature of the C# language. Thus no extra manual work was required when adding or changing the parameters.

As the parameters were only saved on the eBox, the NUC software requested them at every reboot. Thus the parameters were only saved in one place and there was no fear of having different settings on different machines. Also, any time the parameters were changed, the eBox sent a status update to the NUC.

As the p1 parameters were designed to be changeable only when the robot was rebooted, this design was enforced by making the p1 parameters to actually change only during bootup. This ended up causing minor problems, as sometimes the changing of the parameters would not actually require powering down the whole robot. This was the case e.g. when the axle modules were changed. As they needed to be changed quite often during the competition, the eBox was also required to be rebooted quite often.

Online tuning of the parameters was possible using the Remote User Interface, which could connect to the robot over Ethernet or Wi-Fi.

Remote User Inferface (UI)

The on-board computers were configured to run the algorithms and communication software independently, but the user was able to access and configure relevant data on the algorithms, hardware status and parameters on a remote user interface. The remote UI was run on a laptop and communicated via UDP over Wi-Fi or Ethernet.

The user interface of the Remote UI software was divided to different tabs, each of which contained related controls and views. On the left-hand side of the user interface there were views and controls, which were always available. These included the connect button, connection status view, battery status bars, control mode selection, and the commit changes –button to commit new tuned parameters.

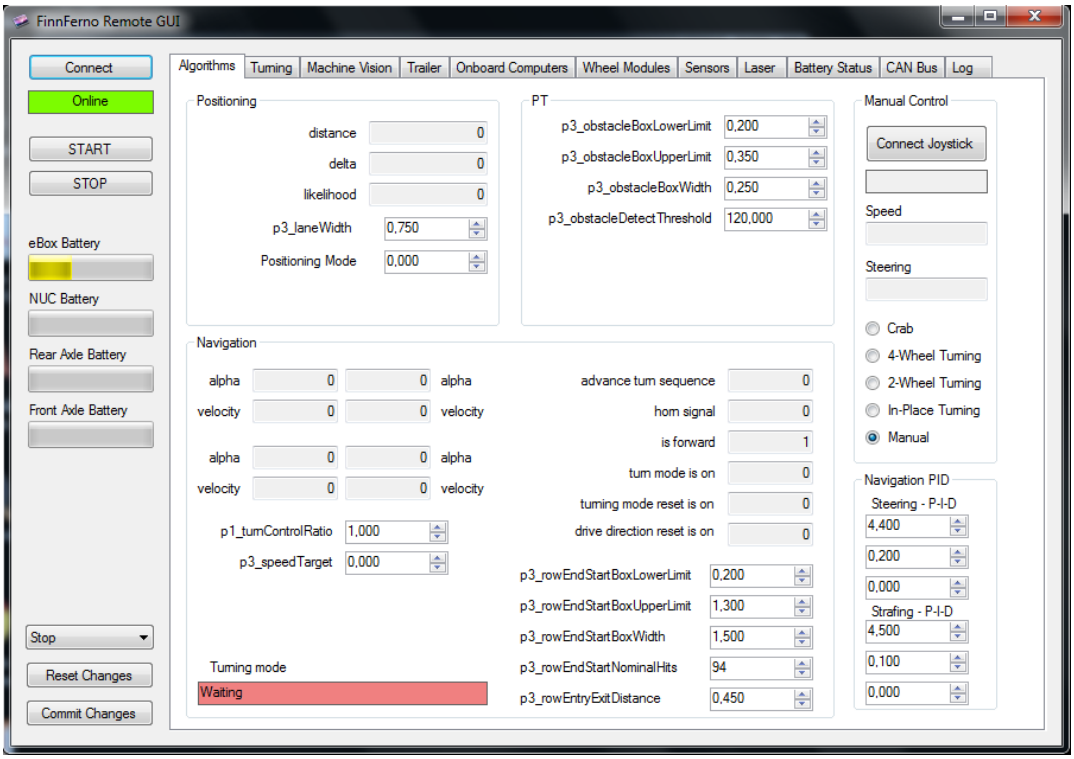


Figure 3.34: *Remote User Interface main window. Different areas of the robot's functions can be accessed with the tabs*

The main purposes of the Remote UI were to tune the parameters, view sensor statuses for debugging purposes, and remote control of the robot.

The remote control of the robot was possible with a standard game controller device. The Remote UI was able to detect and connect the controller and use it to send control messages to the robot.

Sensor statuses were shown and visualized on the Remote UI. This enabled fast debugging of simple problems with the algorithms and the on-board software. The data from RangePacks was shown and visualized for fast evaluation of row detection.

The data from wheel modules was shown, which made debugging and testing the control messages easier. Finally, the laser data was visualized to enable the user to see what the robot sees.

The eBox software tracked the amounts of the received CAN messages, and these counters were shown on the Remote UI. Monitoring and debugging of the CAN communication was faster when using the counters as first indicators of possible problems.

The parameters were shown in number boxes with up and down arrows on the user interface. When the parameter value in the box was changed to different from the one currently in use on the robot, the background of the field would turn red. After committing the changes, the robot would respond with the current parameters in use and the field would turn white again.

During development, the adding of parameter controls to the user interface required no coding. Adding a number field to the user interface and giving it the name of the parameter was enough. The rest of the handling was done using *Reflection* technology, which enabled iterating the parameters by name.

Row Positioning

A Algorithm – Reverse kinematic simulation with distance data

The row positioning A-algorithm uses reverse kinematic simulation to provide the distance deviation from the centre of the crop row, heading deviation from the direction of the row and the likelihood of a successful result at any given time.

The algorithm uses the RangePack modules exclusively, since they provide the most basic measurement data. 20 latest measurements from each infrared and ultrasound sensor are used for calculation to increase the algorithm's robustness, but no other filtering is applied. Although the use of several consecutive measurements does introduce a slight delay in the output, due to the control loop duration the delay is only about 0.8 s. During testing using the simulator the algorithm was proved to be quite accurate, especially at low velocities.

The algorithm works by the idea that when one can locate a plant of an adjacent row relative to the previously measured plants by keeping track of its odometry. It collects 20 of these relative measurements of each sensor. As the robot's measurements were known, the collected measurements could be projected to the robot's coordinate system. This creates a point cloud that would be similar to a very local and time variant map of the plants.

From the local map of plant rows, it is possible to determine that the row curvature is negligible making it possible to approximate them with two lines. The robot's location in the local map can be calculated by analyzing the positions of the lines relative to the origin. As the two lines differ in location, combining them is easier in a previous step:

Instead of creating a two approximated lines, we modify the point map by shifting the points towards the center of the robot, such that all the points form a single line. The best fitting line is then calculated using the general Least Mean Squares algorithm. The parameters *a* and *b* of equation

$$y = ax + b \qquad (1)$$

are selected such that the cost function

$$sum(e\_i\hat{}2) \qquad (2)$$

is minimized, *e_i* being the error of the point *i* from the line's approximated points location.

The parameters *a* and b are defined in such a way that *a* , the steepness of the line conveys the information of the **delta** shows how much deviance there is in the angle of the robot's heading and *b* conveys the information of **d**.

During the Field Robot Event it was found that this algorithm clearly did not work as tested in the development phase. The reason for this was not found conclusively, but it was suspected that in addition to incorrect parameters, the sensor simulation of the RangePack sensor was not an accurate representation of the actual environment.

B Algorithm - HOUGH laser

The B algorithm for positioning was used during the Field Robot Event in Germany. It is based on the Hough transform and feature detection algorithm, which was also used in the machine vision implementation of FinnFerno, described in part 0.

This algorithm proved quite successful despite limited testing, but quite inefficient computationally. This led to some noticeable lag in the positioning data acquisition process.

C Algorithm – Laser scan data segmentation

The row positioning C-algorithm is based on the research of Núñez et al. They have developed a feature extraction algorithm that uses laser scan data to find coherent features, such as walls, corners and curves in the robot's surroundings. They applied adaptive thresholding for the feature inclusion criteria, which accounted for the standard deviation of individual laser scan measurements and the systematic measurement error of the scanner, increasing the algorithm's reliability. (Núñez, et al., 2006)

The algorithm iterates over the entire output of a single scan and determines whether adjacent scan beam results belong to the same feature or not. Here the adaptive threshold is used to compensate for the inaccuracy of the sensor and provide more

accurate data on whether a beam has hit the same feature as the previous beam or not (Núñez, et al., 2006).

For our implementation this algorithm had to be adapted for a non-structured environment of a corn field. The problem is essentially that while using a laser scanner such as the LMS100 indoors produces quite clearly defined features, in an outdoors environment these features are distorted. Thus the data requires preprocessing to find the rows and edges of the corn field. In our algorithm the preprocessing finds segments in the feature data and attempts to combine individual segments into actual features which are also shifted to bring them close to the origin. These features can then be used to determine the displacement and heading error as before in the A algorithm.

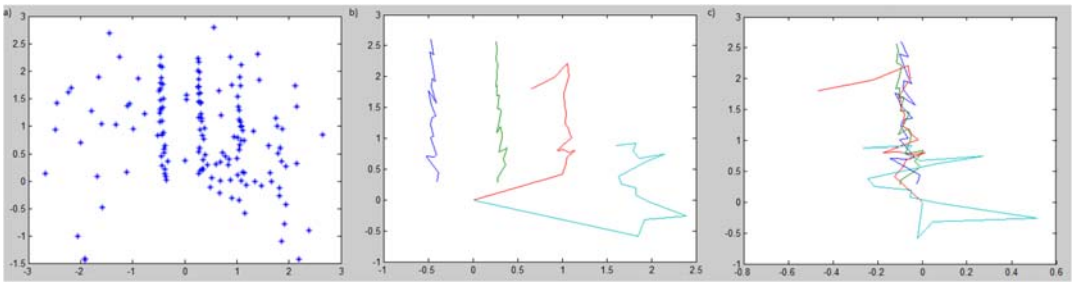


Figure 3.35: *a) Raw laser data, b) segmented laser data, c) segments shifted in the robot's X-axis to center them around the robot.*

The benefit of this algorithm is that it can find features in the current row as well as in the adjacent rows. As has been found in previous years, the LMS100 has a tendency to have some scan beams travel between the leafs of the plants and bounce back from the next row. With this algorithm these “stray” measurements can be utilised to some degree. To reduce the risk of using incorrect features for positioning, the features farther away from the origin are given less weight than features close to the origin.

While this algorithm showed great promise, it was only implemented just before and refined during the Field Robot Event. Thus it was never tested and due to the incompatibility with Matlab generated C-code and the way data was processed in the algorithm it was never used during the event. The problem appeared to be in the variable size of the number of features and points within each feature, which is incompatible with the C arrays of Matlab.

Row Navigation

A Algorithm

The A algorithm for navigation uses an inverse kinematic model to determine required turning angles as a function of the displacement from the center of the row, **d,** and the error in heading from the row direction, **delta**. The inverse kinematic model can be solved analytically, but due to the high degree of freedoms in our robot we used an

empirical approach. Determining the inverse model is done by giving the robot steering angles from minimum to maximum at a constant velocity of 1 *m/s* and measuring the resulting radial and lateral accelerations.

For the heading correction the front and rear wheels are turned in opposite angles to induce turning of the robot. The resulting rotational velocity is measured. For the displacement correction the front and rear wheels are turned in identical angles to induce lateral movement of the robot. The resulting lateral velocity is measured.

Heading correction and displacement correction use two individual PID controllers to determine steering angles to cancel out **d** and **delta.** The results are balanced to a single control signal by linear interpolation of the two separate signals. The balance is determined with a configurable parameter. The heading correction algorithm also determines the Ackermann steering angles for inside turn and outside turn wheels.

The velocity of the robot is determined as a function of the probability of being in a row, the likelihood of correct position data and speed limit of the robot.

Obstacle Detection

In Field Robot Event 2014 Task 2, a robot has to detect obstacles within corn rows and react by reversing back to the headland.

The obstacle detection algorithm uses the LMS100 sensor to map a window in front of the robot. If a set number of points are detected within the window, it is determined that there is an obstacle in front and the robot goes into reversing mode.

Headland Turning

The headland turning algorithm is triggered when the probability of being in the row drop below a set limit. When the algorithm is triggered, the robot is first driven out of the row. After this the algorithm reads the next turn command and proceeds to drive accordingly.

Each command contains the direction of the turn, amount of rows to be driven and the style of the turn. The robot can cross rows either by turning normally or by turning all four wheels in the same direction i.e. crab turning, similar to previous field robots. In addition to these styles, our robot can do a spinning turn, where diagonal wheels are turned to identical angles. This provides a very accurate mode of turning, as there is no need to keep track of the place of the robot during the turn.

When the robot has travelled across the rows it will turn back to the row and drive forwards until the probability of being in the row. At this point the algorithm will give control back to the row navigation algorithms.

Initially the algorithm uses dead-reckoning to drive in the headland, but it was intended that the position in the headland would also be tracked using a more

advanced method. Unfortunately, the turning performance was not tested outside of the simulator, as time ran out trying to solve other problems in the system hierarchy.

Weed Detection

Weed detection uses an ECCI color transform and thresholding to find areas that contain a target colour. These areas are then processed using a Hough transform that is tuned to finding circles. The found circles are then processed and filtered according to size. The final result gives the location of the weed, i.e. golf ball in the picture frame. This can then be easily mapped to world coordinates relative to the robot using the height and angle of the cameras.

Machine vision

Machine vision was used in the robot in two main roles: Positioning and detecting colored golf balls (“weeds”) in Task 3. The robot is designed to carry up to two industrial gigabit Ethernet Pixelink cameras. The cameras can be mounted as a stereo pair or as a monocular vision system. The cameras were mounted on the top of the robot on a rotating platform with adjustable tilt. The cameras’ image is read with PixeLink’s proprietary windows SDK.

OpenCV was be used for image processing, and the machine vision code ran on a separate windows machine (Intel NUC). All of the machine vision code was written in C++ (but the OpenCV C library was used instead of the C++ functions since previous code used the C library)

The intention was to have separate A and B algorithms for tasks 1,2 and 3 (positioning and weed detection), where the B algorithm would have been more sophisticated and A simpler. However, due to time constraints practical field testing time was very limited (one successful run before leaving to Germany), and only one algorithm was developed for both tasks.

Figure 3.36: *Camera mount. The angle of the cameras can be easily adjusted as needed*

Positioning

The machine vision system was used to detect the position of the robot relative to the maize field. This information would then have been used as an input to the navigation algorithms that control the robot's path through the field.

The implemented algorithm used a single camera and basic image processing and Hough transforms to detect the plant rows and hence the robots position.

The ECCI transform

The first step in the image processing was the so called "ECCI transform" (Maksimow, et al., 2007). The ECCI transform is useful in finding a specific color from an image. In the ECCI transform, the RGB color space is projected into a rotated color space where one of the principal vectors is in the direction of the target color, one in the direction of intensity (the white color) and the last one is selected so that the base remains orthonormal.

The following formulas were used for the transform:

$$CEC = XRGB * Vtarget \quad (3)$$

$$Ccross = XRGB * (Vtarget\ VI) \quad (4)$$

$$Cintensity = XRGB * VI \quad (5)$$

where *XRGB* is the RGB color vector for a pixel to be transformed, *Vtarget* is the target color vector scaled to unit length, *VIis* the color vector to the direction of intensity (white color) and *CEC*, *Ccross* and *Cintensity* are the EC, C and I channels respectively.

Thresholding

After the ECCI transform, an adaptive thresholding step was applied. This split the image into two parts: the bright part and the dark part. A percentage slider could be adjusted in the parameters that told the algorithm how many percent of the pixels in the image should end up in the dark parts and bright parts. (For example: Divide the image so that 30% of the pixels end up on bright and 70% dark). This relied on the assumption that while driving inside the row lighting conditions will change, but the approximate amount of maize visible stays fairly constant.

Perspective transform

The image's perspective was then transformed using OpenCV's functions. The purpose was to have parallel lines on the ground plane appear parallel in the image as well.

Hough transform

After this, a Hough transform line finding step was applied. The resulting lines were averaged/grouped, and the resulting average angle and position of the lines was used for approximating the angle and position of the robot relative to the rows. The resulting angle measurements were usually better than the laser scanner algorithm we used. The row center offset measurement turned out to be more difficult, already because the camera mast changes position rapidly from the swinging of the robot.

Object detection

For weed detection the ECCI transform's CEC was used. OpenCV's Hough circle finder was used for finding the brightly colored golf balls from the field. The camera was angled 90 degrees sideways, so we could assume the weed was right beside the robot when it crossed the center line of the image - this made weed signaling and positioning much simpler than with a forward-facing setup.

OpenCV's algorithm worked very well after some smoothing of the image, but the problem was lack of practical testing: since the robot only started moving in the field a week before the competition, we had very little chance to test the system in different lighting conditions. The dataset we had was taken in a very even illumination on a

cloudy day, but during the competition it was sunny and partly cloudy the whole time. Needless to say this caused problems(which was already known), but frankly very little could be done about it because of the lack of data.

Since the “A” algorithm in the object detection worked so well, we didn’t really even consider a “B” algorithm.

In the end the robot detected all the weeds it passed, and had only a few false positives.

Weed killing trailer

The trailer was designed to be used with the main robot in the Freestyle task of Field Robot Event. The goal was to introduce some practical and innovative technology which also has some visual effect for the audience.

During the meetings two main ideas came to the final line and finally a trailer with a flamethrower to kill weeds as thermal plant protection was chosen to be developed. The trailer was designed to be fully operational independently, having its own power source and control systems, and an ability to command the speed of the robot via an ISOBUS-inspired CAN-bus protocol, by following ISO 11783 Class 3 Tractor pattern (a.k.a. ISOBUS TIM). An OpenCV –based machine vision system with two cameras was implemented for the detection of the undesired weeds, and a robotic arm system was designed to target a gas flame towards the weeds.

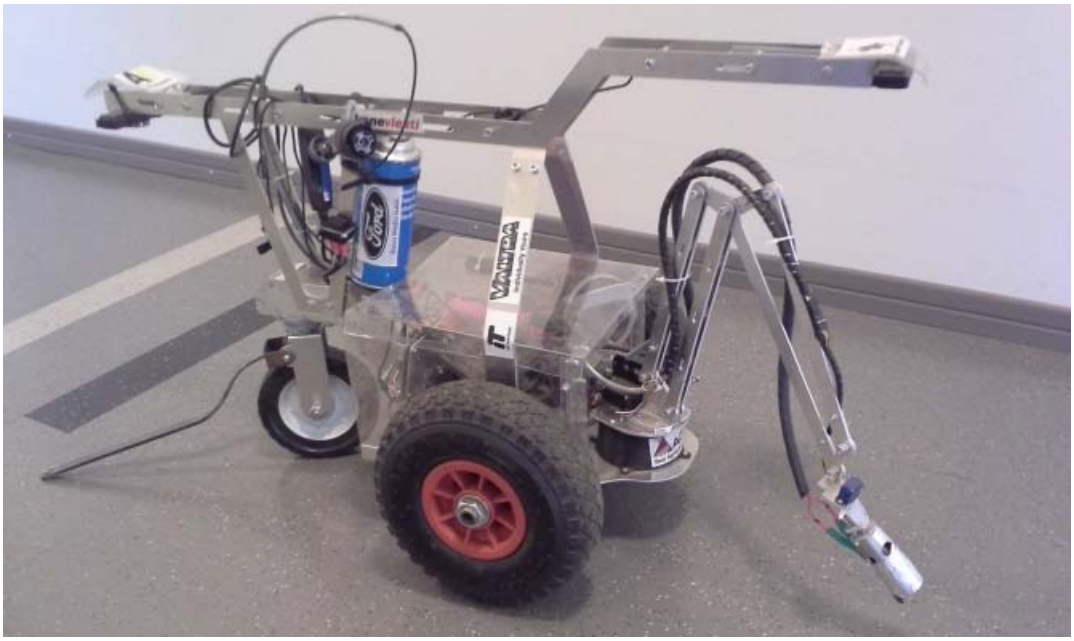


Figure 3.37: *The flamethrower trailer used in the freestyle task.*

Mechanics

The trailer was designed to have two axles to stabilize the movements and to protect the flamethrower arm. The main idea was to use axle structure similar to full trailers (or semi-trailer with a dolly). To reduce the weight of the trailer the front axle was replaced with a single wheel fastened like bicycle front wheel. Lightness and low draught resistance were taken into account in the overall design; aluminium was used as material for the frame, rear axle was drilled hollow and the rear wheels were relatively large to ensure a small rolling resistance.

The trailer frame was designed to provide a suitable platform for all of the electronics on board, as well as the gas equipment and the flamethrower arm. The frame and the flamethrower arm were manufactured from 2-4 mm aluminium sheet. The parts were laser-cut according to the CAD design files, and shaped to the desired form. The frame had also an important function to serve as the stand for the weed detection cameras in the front and rear of the trailer.

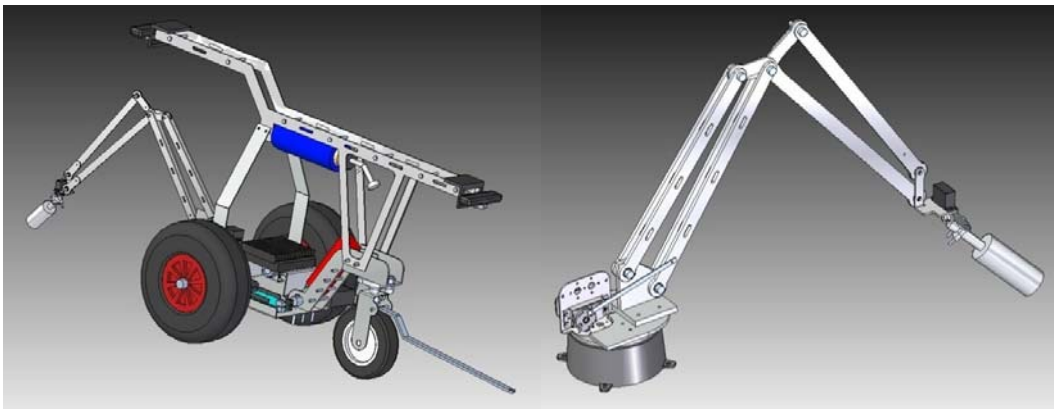


Figure 3.38: *A CAD-drawing of the trailer, illustrating the location of all the essential components, and a detailed illustration of the flamethrower arm.*

The arm was designed to act like a parallel crane introduced in forestry machinery. This structure allows only parallel movement which eases the controlling. Additionally, the arm and the pointer nozzle were able to turn approximately 180 degrees. The nozzle turning was designed to ensure that the flame is pointing always backwards and not flaming the corn plants. The arm was attached to a Lynxmotion Base Rotate Kit rotator platform, and the arm movements were controlled with three servos, one for each degree of freedom: stretching, nozzle turning and platform turning. Streching and platform servos were standard HS-422 servos and nozzle servo type HS-55 microservo.

Flamethrower unit was modified from a handheld weedburner that uses butane as fuel. The ignition spark was produced with a Sparkfun 4.8V Spark Gap Igniter controlled with relay. A solenoid valve with a relay was used to control the gas flow. Two Genius WideCam F100 cameras were used for weed detection. One camera was located in the front of the trailer to initially detect the weeds. The other camera was located in the back of the trailer, above the arm, and it was used to guide the nozzle to the detected weed.

The mechanical design of the trailer proved to be functional as well as reliable, since no mechanical failures or any problems whatsoever occurred during the testing period or in the competition.

Electrics and electronics

The functions of the trailer were controlled by one FitPC computer and one Arduino Mega microcontroller board. The FitPC was used to run the machine vision software for the weed detection and to manage the events during the operation. The microcontroller was used at a lower level to control the arm servos and the igniter and the gas valve relays according to the commands from the FitPC. The communication between the computer and the μC occurred through a serial port. For the communication with the robot, a Kvaser CAN adapter was connected to the FitPC.

The whole trailer was powered with one 14.8 *V* LiPo battery. Power was distributed through a cut-off circuit, identical to the one in the main robot, to 5 *V* and 12 *V* voltage regulators. 12 *V* power was used by the FitPC and the solenoid valve, as the rest of the devices were running on the 5 *V*. A self made PCB was attached to the Arduino board as a shield to make connecting the relays, servos, LED's and power feed easier. The main power switch was connected to the cut-off circuit, while the servos, the gas valve, the igniter and μC had their own switches. The whole electric system of the robot is illustrated in the connection diagram in Figure 3.40.

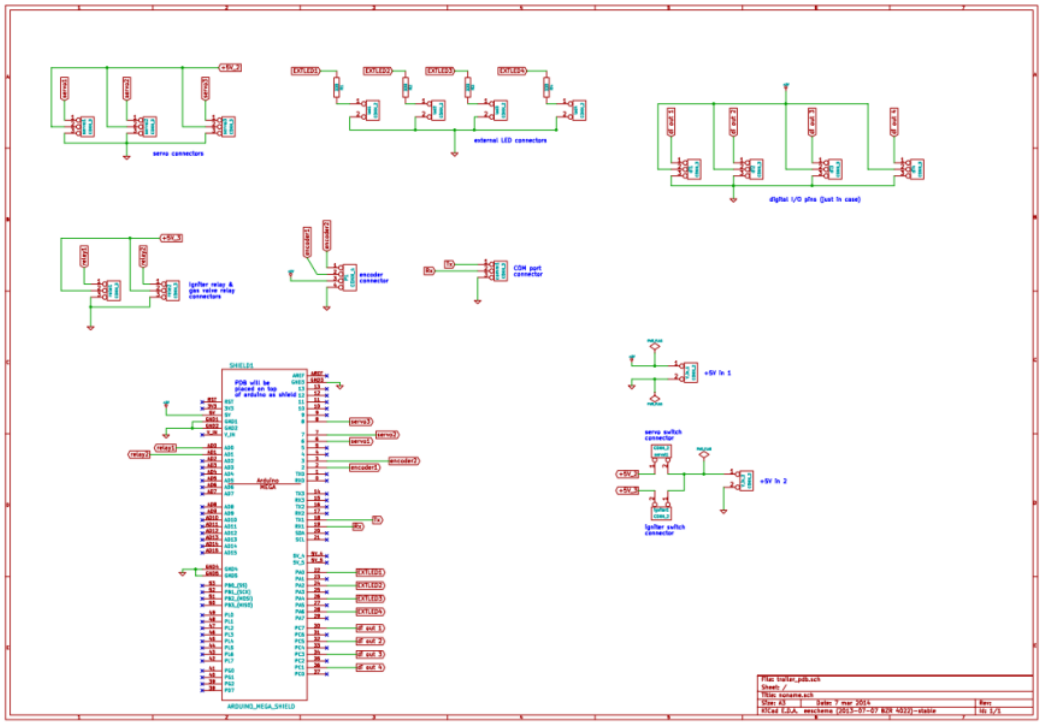


Figure 3.39: *Schema of the designed Arduino Mega shield.*

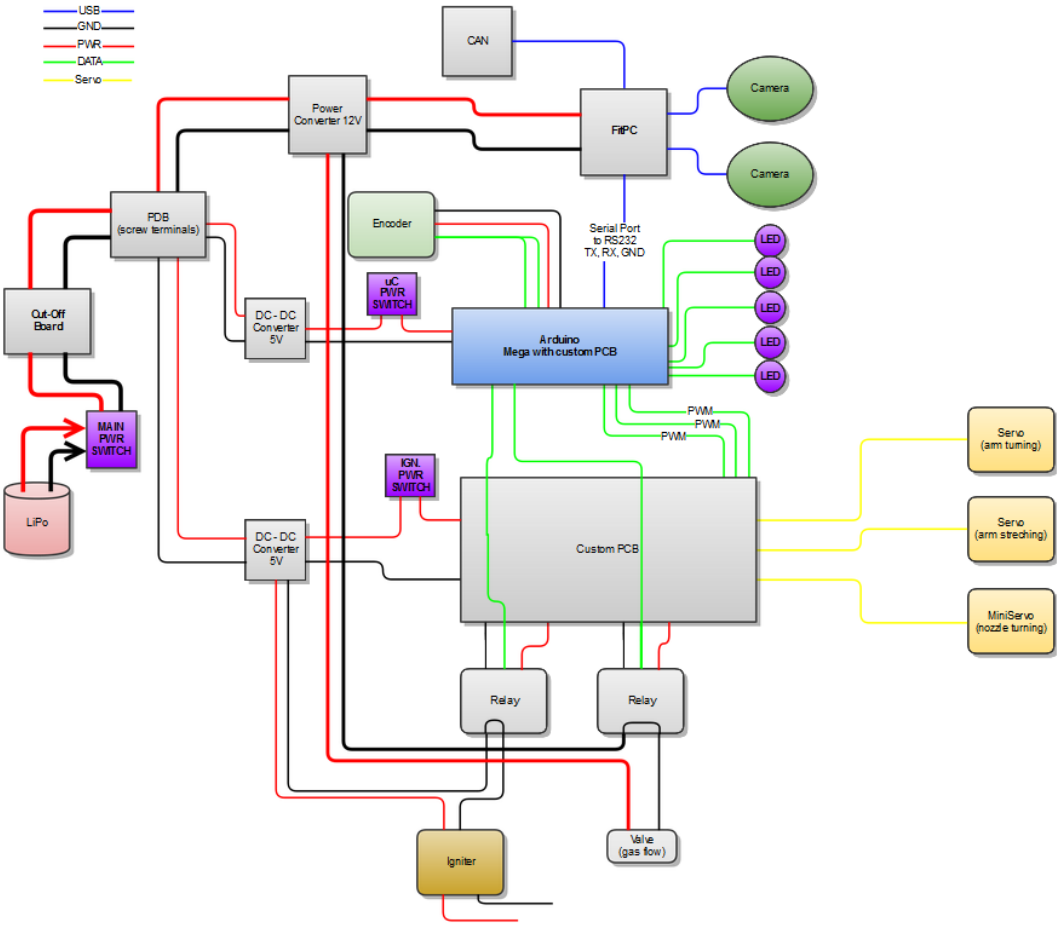


Figure 3.40: *Connection diagram of the trailer.*

Software

The trailer uses two main programming languages. The FitPC computer with Windows XP was running a Python script while microcontroller code was written in C using Codevision AVR.

Python was the highest level code in the trailer. Trailer was run with one main Python script which used separate scripts for machine vision, CAN communication and serial communication. Python was selected as the main language since there were easy-to-use open source computer vision OpenCV libraries as well as serial communication libraries for microcontroller communication available. Kvaser offers CAN libraries for C and C#, so the Python Ctypes library was needed to transform the functions. This setup worked without major problems, apart from some instances of the Python virtual machine crashing.

Machine vision

Machine vision was done with the help of OpenCV tutorials which were freely available online. Machine vision was based on colour recognition which is why deep pink or magenta coloured weeds were chosen for the task. This colour was clearly different

from the colours that normally occur on field conditions which makes the detection dramatically easier. The resolution used was 640x480 and average framerate was between 10 and 15 fps.

Every frame was read and transformed from BGR to HSV color-space. After this the HSV image was thresholded for the magenta range from [148,54,150] to [190,200,255]. From this thresholded image contours of the desired areas were obtained for the area calculation. The size of the desired colour areas in the pictures were determined to avoid false detection. When a desired area was found, the smallest possible circle was fitted around this area and the centre point of this circle was taken as coordinate point (x,y). This coordinate point was then used to guide the arm to the right position.

Figure 3.41: *Sample picture of trailer's machine vision. Left original picture, middle thresholded and right picture with coordinate point obtained.*

Guiding of the arm was done with the coordinate point and eighteen different correlations based on the location of the coordinate which were empirically adjusted. The adjusting coordinate was then transformed into two servo commands specified later.

Trailer Function and Python

Python code was constructed to run in different modes. These modes were determined by the main robot. Trailer was getting modes from 0 to 2 and transformed these into trailers own modes which were from 0 to 3 as illustrated in Figure 3.42. Freestyle task was planned to be run between the corn rows and the robot was supposed to turn for the next row after running to the end of the row. To eliminate the change that trailer starts flaming against the audience it was decided that weeds were placed only between rows and not on headlands. The trailer programming principles were done according to this. Trailer was listening to the modes the robot was sending via CANBUS. When running between the rows the main robot was sending mode 2,

which allows trailer to burn weeds and while robot was turning, the mode was 1 so trailer was waiting.

Trailer mode 0 was equal to stopped mode of the main robot and resulted with no functions and no lit LEDs. This mode was made to use when connecting or disconnecting the trailer from robot when no modes were yet received with the initial idea of automatic coupling. Since this idea was abandoned this mode became less useful but it remained the possibility to let trailer computer run if no modes were coming from the robot. Robot debugging was done using this ability.

Trailer mode 1 was intended to use only when the robot is moving to the start of the row or is performing a turn to the next row. During this mode only the one of the LED, called the StandBy LED was lit as seen in Figure 3.42.

When robot was sending mode 2 it let trailer the full independency to detect and eradicate weeds. In this mode robot was listening and trailer was determining the travel speed. Trailer mode 2 was marked with two adjacent LEDs. During this mode the front camera became active to search weeds as described earlier. When a detection occurred trailer lighted also the third led and trailer mode changed from 2 to 3. After detection trailer changed the wanted speed to 0 m/s and started to search the weed with the back camera. When the weed was found with the back camera all the four LEDs were lit. This mode is the 3 ½ in the Figure 3.42. After detection occurred 25 times coordinate of the weed was taken and the first and the last of LEDs went off. This mode was described as the trailer mode 3, submode 1. During this mode servo commands were sent and the arm was moving to the correct position. Finally when the arm stopped the movement relays of the spark and gas valve were activated and the weed was eradicated. During this mode only the third LED was lit (6).

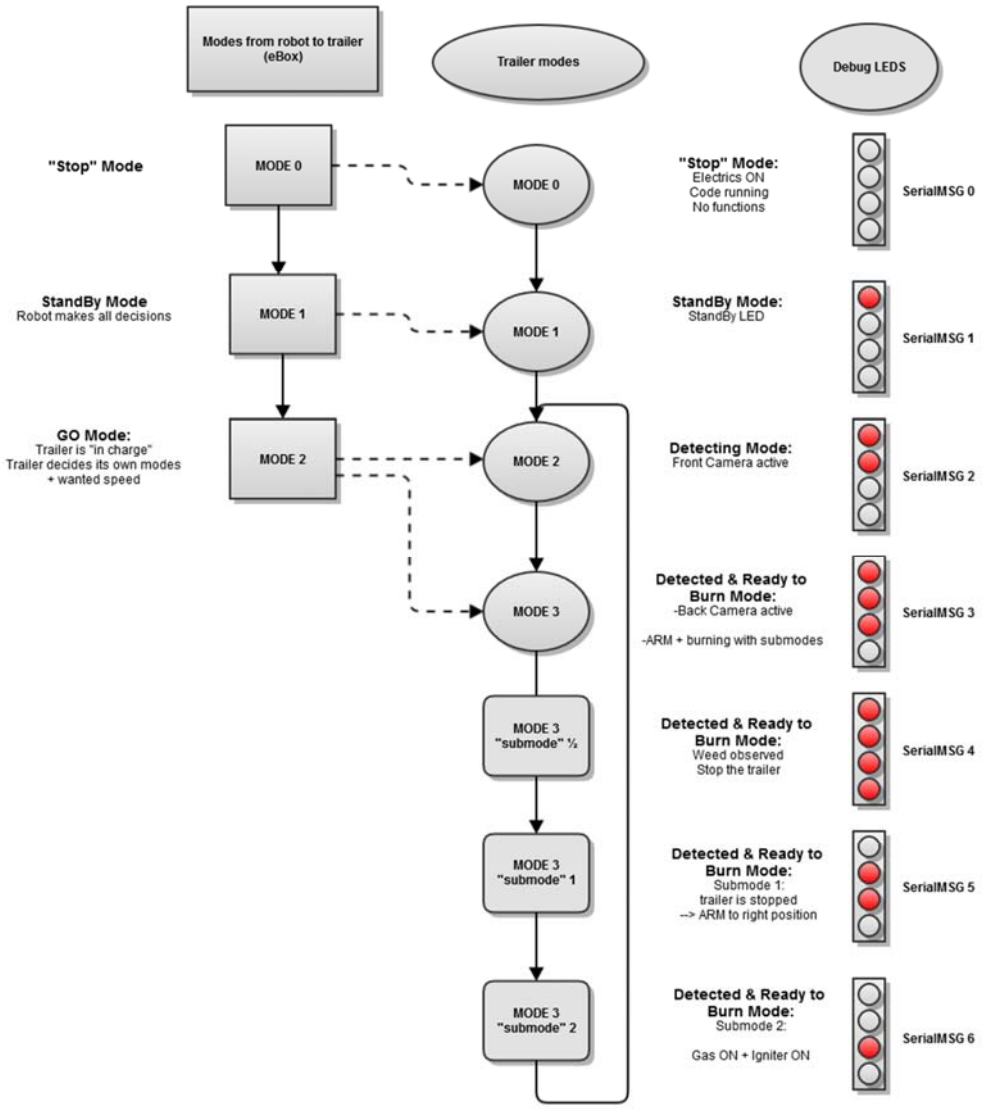


Figure 3.42: *Main principle of trailer software, modes and debugging LEDs.*

Communication and C code

Main communication was done between trailer's FitPC and robot's eBox via CAN-bus. Trailer was receiving the modes, possible errors and actual speed of the robot. Correspondingly trailer was sending its own mode, possible errors and wanted speed when the situation allowed it.

Sending and receiving modes and speed data was the most important part of the communication. Errors were planned also to be used in the message but trailer was not programmed for own errors and was only sending the received error message back. Reason to this was that typically if error was occurring in the trailer the whole program was crashing and was no longer able to send any messages.

Communication between microcontroller and computer was done over RS-232. The python program was sending a message every program loop to the microcontroller. The message structure had a start byte 255, followed with eight command bytes with values ranging from 0 to 254 for servos, debug LEDs and relays. The stretching servo was getting values between 0 and 254. This value was the distance from the back end of the trailer (0 cm) to the maximum reach of 25 cm. The nozzle and rotator turning servos were controlled both with the same value which was ranging from 0 to 180. Value 0 represents the far right corner and 180 far left respectively. All these values were then transformed in the microcontroller into PWM signal individually for each servo. LEDs and relays were simply controlled by sending either 1 for engaged or 0 for disengaged. So the message “255 90 20 0 0 1 1 1 1” was telling to hold the arm in the centre position, approximately 2 cm behind the end of trailer, to hold all the four debug LEDs on and both the relays off.

During trailer tests lots of crashing was occurring, which was suspected to be due to the weak insulation of the igniter cables, causing them to short on the frame of the nozzle. This typically resulted in communication failure between the microcontroller and the computer. Due to this an additional debug LED to signal microcontroller crashing was added and existing debug LEDs were used for communication failure indication in addition to the normal use of mode signalling.

Testing also showed that arm was moving too fast, increasing the risk of breaking the servo. After trying to mechanically suspend the arm with a spring, it became obvious that the slowing needed to be done on software scale. Finally all servo commands were modified with timer after receiving them from the computer. This not only ensured the durability but also made the trailer behaviour more stylish.

Communication between robot and trailer was working well during the competition even though the testing beforehand had been very limited. Only problem was that one of the trailers parameters that was controlling the wanted speed was not adjusted perfectly due to lack of testing time with the main robot and the freestyle performance was not 100% successful. In spite of this the team received good comments on the freestyle implementation.

Test field

Test field for the field robot "FinnFerno" was created on the fields of the University of Helsinki research farm. Test field was prepared by two members of FinnFerno team. Members are PhD. students of Department of Agricultural Sciences.

The field consisted of three times five rows of corn with a row spacing of 0.75 m and ca. 2 m headlands for turning. The test field layout is presented below in Figure 3.43. If the corn should fail, a similar pattern of barley was sowed next to the corn plot.

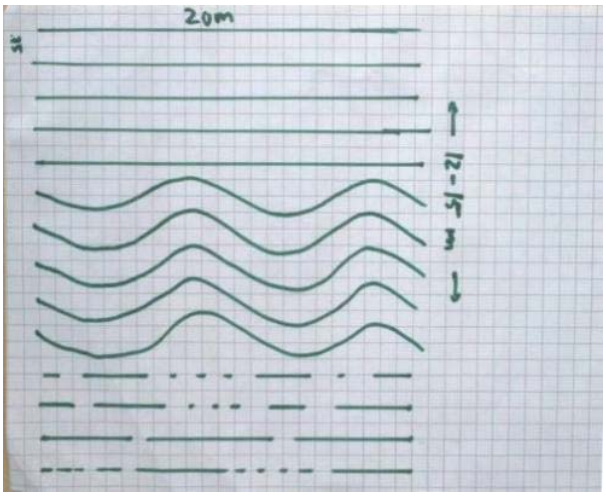


Figure 3.43: *Layout of the test field*

Seed bed preparation: the primary tillage for the field was ploughing, which was conducted in autumn 2013.

Field preparation started very early due the early spring in Finland. Temperature (>22 Celsius degrees) was high at the end of April so field preparation was able to start 22th of April. Levelling of the test field was conducted 22.4. The soil was very loose and no additional secondary tillage was required.

Figure 3.44: *Test field levelling 22th of April 2014*

Corn was sowed on 23.4. using a two row corn seeder, Figure 3.45. Distance between the seeds was adjusted to 15 cm. The test field was fertilized with commercial fertilizer YaraMilaPellon Y-1. Pellon Y-1 consist of nitrogen 27%, phosphorus 2%, potassium 6% and sulfur 3%. Given amount of nitrogen was 300 kg/ha. The corn plot was covered with a gauze ca. one week after sowing to contribute germination and early development.

Figure 3.45*: Machine and tractor for sowing. Sowing was made 23th of April 2014.*

The gauge for covering the corn plot was the same as greenhouses use protect plants. The test field after the gauge was spread is illustrated in Figure 3.46. The use of gauge was a good decision. A few days after the gauge was spread to the corn plot, temperature dropped dramatically in the beginning of May. Totally temperature dropped to 4–6 Celsius degrees in daytime and during night temperature was even below zero.

Figure 3.46*: The test field after the gauge was spread.*

The barley plot was sowed on 1.5. by the research farm staff.

Chemical weed killing was carried out 26.5. (Lentagran WP 3 kg/ha). As the weeds were still going strong ca. one week after spraying, possibly due to the unusually cool weather period, they were removed mechanically on June 2th.

The test field succeeded even though cold weather period existed in May and June. Height of the crop was sufficient for the robot and machine vision testing. At 10th of June high of the crop was ca. 20 cm which is good result for corn in southern Finland. The surface of the test field was also dry and smooth, so the robot was able to move on the field successfully. The ready test field is illustrated in Figure 3.47.

Figure 3.47: *Test field ready for robot testing*

4. Conclusions and Discussion

Implementation Results

Mechanics

Although there are a number of features in the FinnFerno robot that present improvements or alternatives from previous Finnish fieldrobots, our team did encounter a significant number of problems in implementing all of them.

First and foremost among our problems was the new axle module and wheel module design. Since the modular wheel design is such a far cry from the axle modules of the previous years, we found that just the task of making the CAD models took almost 2 months longer than expected. Additionally, the re-design of the controlling electronics and the building process both took longer than was allotted for these tasks. While it would have been preferable that the axle modules had been ready for programming

and stress testing by March, already about one month later than optimal, they were actually ready for driving at the end of April.

Mechanically the wheel modules did not prove very durable, especially in heavy dirt terrain. There were a number of problems that came about during testing and the competition. During testing two wheel modules had the input shaft pinion gear come loose, which resulted in the loss of wheel power. This problem was especially troublesome as it required almost complete disassembly of the wheel module in question.

In Germany, during the preparation for the competition, two wheel modules had their servo axle come loose from the steering transfer wheel, resulting in both the axle of the servo and the nylon fitting of the transfer wheel to wear completely smooth. In another wheel module, the motor input gear was incorrectly aligned, which caused the gear to dig in to the one adjacent to it. With this that particular wheel module was essentially un-usable, as the gear had worn badly, and accurate fitting of the gears would have been very bothersome outside in the field.

Even when the modules were in working order, they required constant maintenance. The steering cables had a tendency to come loose during driving, which would result in the wheel angles to change dramatically, putting additional stress on the drivetrain. We also had trouble attaching the wheels to the axles, caused largely by the single bolt wheel hub attachment and the small width of the bolt. The result was that the all four wheels kept loosening during driving. This problem was brought under control by making individual wheel hubs for each axle, reducing unwanted travel of the wheel with regard to the axle.

Electronics

The electronics on the FinnFerno were implemented quite successfully with only a couple of problems discovered during testing or the competition, most of which were either solved easily or did not affect the robot's functions in a serious way. One rather serious and time-consuming problem came about when the Rangepack sensors were being integrated to the rest of system.

The problem was detected when more than three devices were connected to the CAN network, which caused the entire network to fault. The fault caused all communication to stop in the network, and by using an oscilloscope to monitor the CAN bus it was found that even the voltage levels of the bus fluctuated widely after plugging in a device to the bus.

Apparently the use of metal standoffs created a ground loop in the RS-485 bus and caused a deviation in the CAN bus voltages. After replacing the metal standoffs with nylon, the problem was solved. Unfortunately one sensor module was damaged during the troubleshooting and had to be replaced.

The NUC PC also presented a challenge with regard to the power supply. As the NUC has a 19 *V* input voltage requirement, a car charger that could provide this voltage had to be found. In the end we were able to find a charger that fit our needs, but due to the use of Li-Po batteries we experienced a number of times when the charger went to a safe-mode state and refused to turn on or output the correct voltage. We could find no cause for this and the only fix seemed to be to disconnect the charger and the output connector and wait a few minutes. After this the power adapter recovered to normal operation.

Algorithms

The algorithms were developed quite early in the project, with the simulator running by the end of 2013. In the simulator they functioned satisfactorily and were left to wait actual testing, when the other parts of the robot would be ready to accommodate the algorithms.

As it turned out, we were never able to actually test the algorithms before the Field Robot Event in Germany. This was caused by several factors, discussed above. As for the algorithms, during the event it was discovered that our simulator did not actually correspond with the actual environment. This caused some confusion, as the algorithms, both positioning and navigation, gave very different output from expected. The A positioning algorithm, for instance, provided a biased output unlike that of the simulator, which caused the robot to constantly turn to the left. This was solved by limiting the measuring distance of the RangePacks to only 25 *cm.*

There were also significant problems in trying to tune the navigation PID controllers, as there seemed to be significantly more latency in the output of the navigation algorithm compared to the positioning algorithm's output. This seemed to cause wild oscillation in the robot's trajectory and caused yet another point of confusion in the on-site debugging of the robot. When a work-around was used, it was discovered that a rapid direction change would cause the turning servos to shear their axles, as discussed above.

Parameter handling worked quite well, although some issues with the parameter updating process were discovered just before the event. These were solved, but displayed the convoluted structure of the Matlab – C# interface. Unfortunately, outside of using another toolset, this is an issue that can be dealt with by acknowledging it well in the beginning of the project.

The Remote UI was one of the best features of our architectural implementation. Thanks to some clever use of the Reflection property in C# and interest of the person responsible for this functionality to get informed in the C# features, new tuneable parameters can be added to the UI simply by adding a numerical field and assigning it with the name of the parameter. This new parameter would then be linked to the parameter structure on the eBox automatically during parameter commits.

Competition Results

Due to the issues discussed above, we were not able to get very good results in the main competition tasks. All in all we achieved 10th place in the overall competition out of 21 teams. This does illustrate the difficulty that is involved in a project as involved as this. There are various ways for dealing with the challenges of the Field Robot Event and as it was with a number of teams, the simplest mechanical solution is easiest to get moving, leaving more time to develop the intelligence of a mobile robot.

In the Freestyle Task our robot was awarded 2nd place, with merely a fraction of point behind the other Finnish entry, the Agro-Bot. This success was largely thanks to the trailer functioning well during the task, and the last minute work done by the team to get the robot moving with the trailer.

Project Management

In terms of the project itself, after the end of 2013 the schedule started slipping by a little at a time, which led at the end of March to a situation where the axle modules were still under work, and the computer system had not been implemented yet. At this point some members of the team were under stress from other courses and projects, and were not able to work on the robot.

Most of the design guide lines and standards in the project were done using specification groups, while certain bigger subassemblies or algorithms were given to working groups. These groups were given a certain goal and possible limitations, and the group would meet together to accomplish their goal. The idea behind these groups was two-fold: Firstly, it would allow members from different disciplines to interact in a common task. Group size would range from two to four, so subtasks would be easier to assign. Secondly, it would divide the work in to smaller and more manageable parts. The goal was to keep specification groups open for one to two weeks, and working groups for no more than two months, according to the approximate of the group's work load.

The management of the project could have been more organised and more care put into making sure that work that had been reported finished was in fact so. At times the team captain could also have put more effort into coordinating work even further. Communication within the team was conducted using *Flowdock,* which made it quite easy to establish smaller groups within the bigger group for more targeted communication. The software itself performed well and it could be utilised more efficiently if the team members make sure to check their feeds at regular intervals.

By the end of the Field Robot Event we suspected that the project would not have needed more than a few weeks of concentrated and synchronised work to get the robot to a state where it would have worked as intended, if not perfectly then at least tentatively.

Some members of the team felt that there was an inherent loss of interest at the very beginning of the project due to the fact, that a number of working methods and tool chains were predetermined from the very beginning. The limited amount of team participation into the selection of tools was given, but more information on why a specific programming environment or operating system would be used during the project.

Delegation of duties proved to be more difficult than expected. At several points of the project some members were handling one or more tasks at a time, some of which were time-consuming on their own, let alone simultaneously. Coordinated work between team members, in a single location would have been very beneficial but we encountered some problems in trying to arrange these times. This issue should be addressed from the very beginning during the learning period in the Fall semester.

Discussion

Overall the field robot project provided a unique experience in product development, as nearly all options are left open to choose from. However, for future project groups it should be advised, that taking on more challenge than is necessary can start wearing on working morale and team morale as the project reaches its mid-point and onwards.

As for the design and construction of the robot it should be noted that time and money could be saved by using better methods to manufacture certain parts of the robot. For the rims, for example, milling each rim from a solid piece of PE plastic was quite wasteful, with more than 50 % of the material lost in waste. In the future a composite design, made up of separate pieces should be considered. The axle modules could also have been designed with thinner plates, as the 4 *mm* plates used in the robot increased the weight of each axle unnecessarily. We suspect that 2 *mm* plate would have been sturdy enough for the robot. This could be a point for future teams to investigate during their design process.

With regard to the software development and testing, and the difficulties with the simulator, it should be noted for new implementations of a simulator that the simulation of the Rangepacks deserves further study. Specifically, we found that in order for the A positioning algorithm in part 0 to work in the real world, we had to limit the measuring distance of the Rangepacks in the software to only 25 *cm* from the original setting of 50 *cm.* The original setting worked very well in the simulator, because most of the simulated pings found their target in the first row of plants, thus providing concise results over time. This was confirmed visually during the algorithm's trial runs in the simulator. In the real world however, the robot tended to stray to either side quite soon after setting of. We believe that this was caused by incorrect measurements, which corrupted the validity of the calculated centreline and heading. This has something to do with the probability of consecutive pings not hitting a plant in the adjacent row, instead travelling to the next row or taking a false positive from the

ground behind the adjacent row. Future teams should make sure they have their sensor systems available as soon as possible, so that adequate validation of the simulator can take place.

For us the development of a novel axle module implementation proved too taxing on the team, and in part led to delays with the rest of the project. Of course, this is not the sole reason for delays in the project: The computer system should have been finished much earlier than it was, so that sensors and interfaces could have been tested before the robot was finished mechanically.

The specification and working groups were an effective work division method, at least in principle. Dividing the work into smaller sections gave the members a much more appealing goal to work towards. However, especially the specification groups need more instruction at the outset, as with certain groups it proved very difficult and discouraging to try and define a standard or specification. This problem could be addressed by giving more information about the principles of standardising.

The fact that some members felt discouraged from the very beginning is a problematic issue. As there was no tangible, professional cause for committing to the project, it is paramount that all members of the team find a personal reason to work on and contribute to the shared workload. We believe that this would be easiest to achieve by giving students choice in the selection of working methods and tools, as was done with our team, but with even more attention put in to the instruction of their use, benefits and risks.

Our team found that some aspects of the design process were very difficult to grasp until the very end of the project. This represents an important point to consider within future Field Robot teams. If some component is difficult place in the project workflow, team members must bring this forward to the instructors, who are then able to help the team more efficiently. Team members should also strive towards better communication within the team, especially if one feels that they have a problem. Ideally all members should be aware what the others are working on, what's their current status and what are their problems within their tasks.

Acknowledgments

Building our robot and the trailer as well as participation in the Field Robot Event 2014 in Strenzfeld, Germany was made possible by our sponsors: AGCO, Henry Ford foundation, Koneviesti, Laserle, Murata and Valtra. Additionally we would like thank Aalto University and University of Helsinki. Finally, we owe thanks to our instructors, Timo Oksanen, Jari Kostamo and Matti Pastell for their support and guidance during this process.

5. Bibliography

Chip45, 2014. *chip45.com.* [Online]

Available at: http://www.chip45.com/

[Accessed 17 8 2014].

Kemppainen, T. & al, e., 2009. *Easywheels Final Report - Aalto University Department of Automation.* [Online]

Available at:

http://autsys.aalto.fi/en/attach/FieldRobot2009/EasyWheels_finalreport.pdf

[Accessed 27 7 2014].

Maksimow, T. et al., 2007. *Department of Automation and Systems Technology - Wheels of Cornture.* [Online]

Available at:

http://autsys.aalto.fi/en/attach/FieldRobot2007/FRE2007_WheelsOfCorntune.pdf

[Accessed 18 8 2014].

Núñez, P. et al., 2006. *Feature Extraction from Laser Scan Data Based on Curvature Estimation for Mobile Robotics.* Orlando, s.n.

Ryynänen, J. & al, e., 2011. *Cornivore Final Report - Aalto University Department of Automation.* [Online]

Available at: http://autsys.aalto.fi/en/attach/FieldRobot2011/Cornivore_FinalReport.pdf

[Accessed 27 7 2014].

Filename: Proceedings_FRE2014-hwg.docx
Directory: C:\Users\Hans\Documents\01-text-data_UHOH\FieldRobotEvent\2014\Booklet and Proceedings
Template: C:\Users\Hans\AppData\Roaming\Microsoft\Templates\Normal.dotm
Title:
Subject:
Author: Reviewer
Keywords:
Comments:
Creation Date: 4/1/2015 4:16:00 PM
Change Number: 62
Last Saved On: 4/2/2015 11:52:00 AM
Last Saved By: ano
Total Editing Time: 103 Minutes
Last Printed On: 4/2/2015 11:52:00 AM
As of Last Complete Printing
Number of Pages: 224
Number of Words: 47 941 (approx.)
Number of Characters: 273 264 (approx.)